Alkulukutesteistä

Lauri Sormunen

Matematiikan pro gradu

Jyväskylän yliopisto
Matematiikan ja tilastotieteen laitos
Kevät 2016

Tiivistelmä: Lauri Sormunen, *Alkulukutesteistä* (engl. *On primality tests*), matematiikan pro gradu -tutkielma, 71 sivua., Jyväskylän yliopisto, Matematiikan ja tilastotieteen laitos, helmikuu 2016.

Tämän tutkielman tavoitteena on esittää tunnetuimmat alkulukutestit niin matemaattiselta perustoiltaan kuin käytännön toteutuksiltaan ohjelmakoodin muodossa. Alkulukutestit jaotellaan yleisesti deterministisiin ja probabilistisiin testeihin; deterministiset testit antavat täysin varman vastauksen, mutta ovat suurille luvuille huomattavasti probabilistia testejä hitaampia. Probabilistiset testit ovat nopeita tehdä, mutta saattavat antaa väärän vastauksen. Testien suoritusaikaa mitataan karkeasti niiden suorittamiseksi vaadittavien laskutoimitusten lukumäärällä.

Tutkielmassa käsitellään deterministisistä testeistä jakolaskumenetelmä, Wilsonin lause, Prothin testi, Lucasin ja Lehmerin testi ja viimeisenä suhteellisen uusi AKS-testi. Yleisesti deterministiset testit eivät ole kovin käyttökelpoisia, sillä niiden suoritusaika kasvaa eksponentiaalisesti testattavan luvun kasvaessa tai ne toimivat ainoastaan tiettyä muotoa oleville luvuille. Probabilistisiin testeihin päädytään hyödyntämällä Fermat'n pientä lausetta käänteisesti, ja tätä kutsutaan Fermat'n alkulukutestiksi. Kyseinen testi ei kuitenkaan toimi kovin luotettavasti: on olemassa Carmichaelin lukuja, jotka hyvin todennäköisesti toteuttavat Fermat'n pienen lauseen yhtälön, vaikka ovatkin yhdistettyjä lukuja. Fermat'n alkulukutestin johdannainen Millerin ja Rabinin testi on kuitenkin erittäin käytetty sen vuoksi, että tälle pystytään määrittelemään virheraja, jolla testi antaa vääriä vastauksia. Solovayn ja Strassenin testi on hyvin lähellä kahta viimeksi mainittua testiä, muttei ole aivan yhtä luotettava kuin Millerin ja Rabinin testi.

Alkulukutestiä, kuten monia algoritmeja yleensäkin, nimitetään polynomiaikaiseksi, jos siinä vaadittavien laskutoimitusten lukumäärä on polynomi testattavan luvun numeroiden lukumäärän suhteen. Pitkään alkulukutestaus ei ollut polynomisessa ajassa ratkaistava ongelma, ennen kuin vuonna 2002 julkaistiin AKS-testi, joka on ensimmäinen deterministinen ja polynomiaikainen alkulukutesti. Tutkielmassa todistetaan lopulta myös tämän testin toimivuus sekä todetaan, että kyseessä todellakin on polynomiaikainen algoritmi.

Sisältö

Johdanto

Tässä pro gradu -tutkielmassa perehdymme *alkulukutestaukseen* ja siinä yleisesti käytettyihin menetelmiin. Tiivistettynä yhteen lauseeseen alkulukutestauksella tarkoitamme positiivisen kokonaisluvun testaamista siltä varalta, että se on alkuluku. Tutkielma on suoraa jatkoa kandidaatintutkielmalleni *Aritmetiikan peruslauseen toteutuminen laskennallisesti*, jossa käsittelin lyhyesti ja yksinkertaistetusti eräitä yleisimmistä alkulukutestejä, kuten Fermat'n testin ja Millerin ja Rabinin sekä Solovayn ja Strassenin testit. Kandidaatintutkielmassani sivusin myös lukujen tekijöihinjakoa, mutta tämän tutkielman painopiste on pelkässä alkulukutestauksessa.

Alkulukutestit ovat karkeasti jaettavissa *probabilistisiin* ja *deterministisiin*, joista ensimmäisen tyyppiset eivät anna täysin varmaa vastausta, vaan "todennäköisesti oikean" vastauksen. Deterministiset testit puolestaan antavat aina tarkasti oikean vastauksen, mutta nämä ovat vähemmän käytettyjä *aikavaativuutensa* vuoksi. Aikavaativuus liittyy oleellisesti siihen, miksi tutkielman aihetta on edes tarpeellista käsitellä: jo peruskouluikäisinä olemme oppineet jakamaan lukuja tekijöihin, ja jossain vaiheessa tätä prosessia olemme huomanneet, että jokin tekijöistä ei enää jakaudu pienempiin tekijöihin, joten kyseisen luvun on oltava alkuluku. Tämä on tavallaan esimerkki yksinkertaisimmasta deterministisestä alkulukutestistä: jos luku ei ole jaollinen millään itseään pienemmällä luvulla, niin sen on oltava alkuluku. Tämä ei kuitenkaan ole käytännön sovellusten kannalta riittävän nopea keino, sillä tehtävien jakolaskujen määrä kasvaa eksponentiaalisesti testattavan luvun numeroiden lukumäärän suhteen, eivätkä nopeimmatkaan tietokoneet kykene tällä keinolla antamaan tulosta järjellisessä ajassa. Esimerkiksi salausavainten luontiin saatamme tarvita vähintään tietyn mittaisia, isoja alkulukuja, joita tällainen yksinkertainen testi ei tunnista kovin nopeasti. Selvitämme tarkemmin paitsi käsitteitä, myös yksinkertaisia testejä luvuissa yksi ja kaksi.

Tunnettujen deterministisen testien hitauden vuoksi runsaassa käytössä ovat probabilistiset testit, jotka ovat huomattavasti jakolaskumenetelmää nopeampia ja antavatkin nykytietokoneilla suoritettuina vastauksen millisekunneissa vielä yli satanumeroisille luvuille. Käsittelemme näitä testejä eräiden muiden yleisten ja nopeiden alkulukutestien rinnalla luvussa kolme.

Nopean deterministisen testin löytäminen on osoittautunut yllättävän vaikeaksi, ja pitkään valloilla olikin luulo, että alkulukutestaus ei ole *polynomisessa ajassa* - eli karkeasti ilmaistuna kohtuullisessa ajassa - suoritettava toimenpide; vuonna 2002 intialaiset tietoteknikot Agrawal, Kayal ja Saxena kuitenkin julkistivat polynomisessa ajassa toimivan deterministisen AKS-alkulukutestin, jonka käsittelemme viimeisenä

tutkielmassa. Sen sijaan esimerkiksi *eksponentiaalisessa ajassa* suoritettavat toimenpiteet ovat suuremmilla syötteillä käyttökelvottomia. Kokonaislukujen tekijöihinjako on esimerkki ongelmasta, johon ei ole kyetty ainakaan vielä löytämään polynomisessa ajassa toimivaa menetelmää.

Käytännönläheisyytensä vuoksi tutkielma sisältää myös toimivaa C#-ohjelmakoodia, jonka kopioimalla on mahdollista testata myös itse testien nopeuksia. Testeistä erityisesti AKS-testiä on mahdollista nopeuttaa radikaalisti ohjelmoimalla kertolaskut toimimaan nopeaan Fourier-muunnokseen perustuvalla menetelmällä; juuri kerto- ja jakolaskujen suorittamiseen käytettyjen menetelmien nopeus aiheuttaa yleisesti ottaen algoritmien suoritusajoissa pieniä eroavaisuuksia eräisiin lähdemateriaalien arvioihin. Algorimien suoritusajat ovat kuitenkin suuntaa antavia ja ilmoittavat lähinnä sen, mitä luokkaa niiden nopeus on suhteessa toisiinsa.

LUKU 1

Johdatus alkulukutestaukseen

Tässä kappaleessa selvitämme syitä alkulukutestauksen tarpeellisuudelle sekä muistelemme aiemmin opittuja tuloksia. Hyvät lukuteorian ja algebran esitiedot tulevat olemaan tarpeen, ja tietotekniikan ja ohjelmoinnin perustuntemuksella tekstistä voi saada vieläkin enemmänkin irti.

Oletamme jatkossa, että luonnollisten lukujen joukko on

$$\mathbb{N} = \{1, 2, 3, ...\},$$

eli nolla ei ole luonnollinen luku. Lisäksi olkoon $\mathbb{P}$ kaikkien alkulukujen joukko. Muut tutkielmassa käytettävät merkinnät ovat standardimerkintöjä ja ovat listattuna sivulla 67.

1.1. Motivaatio

Alkuluvut ovat tunnetusti pitkään kiehtoneet matemaatikkoja ja ovat erityisesti matemaattisessa mielessä erittäin hyödyllisiä. Ne ovat varsin oleellisessa roolissa lukuteoriassa, jossa tutkitaan erityisesti kokonaislukujen ominaisuuksia. Esimerkkejä lukuteorian oleellisista aiheista ovat lukujen merkitseminen sekä niinkin yksinkertainen asia kuin kokonaislukujen jaollisuus: historiallisesti ihmisillä on iät ja ajat ollut tarve jakaa asioita tasaisiin pienempiin joukkoihin. Tässä mielessä on hyödyllistä tunnistaa, mitkä luvut ovat jaettavissa milläkin luvuilla. Jo antiikin kreikkalaiset tunnistivat, että kokonaisluvut ovat luokiteltavissa esimerkiksi parillisiin ja parittomiin lukuihin, sekä alkulukuihin ja yhdistettyihin lukuihin. Esimerkiksi Eukleides käsittelee kirjansa Alkeet seitsemännessä osassa juuri jaollisuutta ja alkulukuja [**7**, s. 193].

Alkuluvut pysyivät pitkään puhtaasti matemaattisena ilona ilman erityisen merkittäviä käytännön sovelluksia, mutta Ron Rivestin, Adi Shamirin ja Len Adlemanin vuonna 1977 keksimä RSA-salausalgoritmi muutti asian: algoritmia käyttäen pystyttiin ja pystytään yhä salaamaan lyhyitä viestejä. Esimerkiksi huomattava osa Internetissä tapahtuvasta tiedonsiirrosta on nykyään suojattu RSA-pohjaisella salauksella. Algoritmin voima perustuu klassisen *aritmetiikan peruslauseen* asetelmaan:

LAUSE 1.1 (Aritmetiikan peruslause). *Jokainen ykköstä suurempi luonnollinen luku voidaan yksikäsitteisesti järjestystä lukuun ottamatta ilmoittaa alkulukujen tulona.*

Kyseinen lause esittää selvästi, että kokonaisluvut voidaan jakaa alkulukutekijöihin. Lause ei kuitenkaan ota kantaa siihen, mitä nämä alkulukutekijät ovat; itse asiassa alkulukutekijöiden löytäminen yleiselle, suurelle kokonaisluvulle on osoittautunut hyvin työlääksi.

RSA-algoritmi hyödyntää kokonaislukujen tekijöihinjakamisen vaikeutta: siinä etsitään kaksi sattumanvaraista - ja riittävän suurta - alkulukua p ja q, joita käyttäen salataan viesti. Näistä lasketaan tulo $n = pq$, josta tulee salatun viestin *julkisen avaimen* osa. Mikäli ulkopuolinen saa selville, mitä ovat alkuluvut p ja q, voi hän purkaa salauksen. Algoritmi toimii seuraavasti:

(1) Etsitään satunnaiset alkuluvut p ja q.
(2) Lasketaan tulo $n = pq$.
(3) Lasketaan Eulerin φ-funktion arvo $\varphi(n) = (p-1)(q-1)$.
(4) Etsitään "lyhyt" luku e siten, että lukujen e ja $\varphi(n)$ suurin yhteinen tekijä on 1. Etsitään tälle käänteisluku d modulo $\varphi(n)$, eli pätee $ed = 1 \mod \varphi(n)$.
(5) Muutetaan salattava viesti kokonaisluvuksi m. Korotetaan viesti m potenssiin e modulo n, jolloin salattu viesti on luku $c = m^e \mod n$.

Tällöin viesti m on salattu luvuksi c, josta helposti voimme laskea alkuperäistä viestiä vastaavan kokonaisluvun korottamalla luvun c potenssiin d modulo n, sillä

$$c^d \equiv m^{ed} = m^{1+k\cdot\varphi(n)} \equiv m \cdot (m^{\varphi(n)})^k \underset{(1)}{\equiv} m \cdot 1^k = m \mod n,$$

missä kongruenssi (1) on seuraus Eulerin lauseesta. Algoritmissa julkisen avaimen muodostaa lukupari (n, e), jonka avulla voimme salata minkä tahansa kokonaislukumuotoisen viestin, ja yksityinen avain on luku d, jonka päättely on vaikeaa ilman tietoa käytetyistä alkuluvuista p ja q.

Selvennyksenä julkisen ja yksityisen avaimen voi ajatella postilaatikon luukkuna ja sen lukon avaimena: kuka tahansa voi laittaa kirjeen (viesti) postilaatikkoon (jolloin viesti salataan), mutta vain avaimen (yksityisen avaimen luvun d) haltija voi nähdä kirjeen (eli alkuperäisen viestin) - ainakaan ilman huomattavaa voimankäyttöä (etsitään tulon n tekijät) tai tiirikoimatta lukkoa (laskemalla luku d muin keinoin, jos salaus on hutiloitu).

Miten alkulukutestaus sitten liittyy tähän? Sattumanvaraiset alkuluvut p ja q eivät löydy tuosta noin vain, vaan ne täytyy etsiä muiden lukujen seasta. Mitä nopeammin alkuluvut löydetään, sitä nopeammin ovat viestit salattavissa RSA-algoritmia ajatellen. Toisaalta alkulukujen nopea löytäminen voi auttaa myös murtamaan salatut viestit helpommin.

1.2. Tietotekniikan rooli

Tarkasteltavien ongelmien luonteen vuoksi tutkimme alkulukutestausta osin tietoteknisestä näkökulmasta. Matemaatikolle alkulukutestauksella ei välttämättä ole suurta merkitystä; matemaatikko voisi hyvinkin olla tyytyväinen todistettuaan lauseen, jonka mukaan ratkaisu on olemassa, eli tässä tapauksessa luku voidaan osoittaa alkuluvuksi tekemällä tietyt - mahdollisesti hyvinkin runsaasti laskutoimituksia - vaativat toimenpiteet.

Käytäntöön sovellettaessa tämä ei kuitenkaan riitä: nopeimmillakaan tietokoneilla ei joissain tapauksissa ehdi osoittaa lukua alkuluvuksi, vaikka käytettävissä olisi

kaikki maailman aika. Tarvitaankin siis tehokkaita algoritmeja - ja parasta algoritmit on kuvailla nopeasti käytäntöön sovellettavana ohjelmakoodina.

Tässä tutkielmassa kuvailemme sanallisen selityksen lisäksi osan algoritmeista suoraan käytäntöön sovellettavana ohjelmakoodina ja osan useille ohjelmointikielille muunnettavissa olevana pseudokoodina. Ohjelmakoodi on kirjoitettu C#-ohjelmointikielelle (engl. *C-Sharp*), joka on C/C++- sekä Java-kielten jälkeläinen. Jos ymmärtää näitä ohjelmointikieliä, niin C#-kielisen koodin lukeminen ei ole vaikeaa. Java ja C# ovat itse asiassa niin samankaltaisia, että käytetty koodi kääntyy lähes muutoksitta Java-kielelle.

Tietokoneiden hyödyntämisen kannalta haasteita asettaa tietokoneen laskukapasiteetin äärellisyys: vaikka tämänhetkiset (vuonna 2015) kotikoneet suorittavat miljardeja laskutoimituksia sekunnissa, ovat käytettyjen lukujen koot hyvin rajallisia. Prosessorit yleisimmin pystyvät laskemaan kerralla vain kahden 32- tai 64-bittisen luvun summan tai tulon, eikä tulos voi olla suurempi kuin prosessorin käyttämät 32- tai 64-bittiset luvut. Koska alkulukutestauksessa on kyse huomattavasti tätä kokoluokkaa suurempien kokonaislukujen testaamisesta, täytyy käyttää lukujen tallentamiseen ja niillä laskemiseen sopivaa tietorakennetta. C#- ja Java-kielistä löytyy *BigInteger*-tietorakenne, jota hyödynnämme tämän tutkielman ohjelmakoodeissa ja joka soveltuu hyvin pitkien kokonaislukujen tallentamiseen. Laskutoimitukset ovat kuitenkin BigInteger-datalla hieman hitaampia kuin tavallisilla kokonaislukutyypeillä (esimerkiksi *int*), sillä lukuja täytyy käsitellä lyhyemmissä pätkissä sen sijaan, että koko laskutoimitus tapahtuisi samanaikaisesti.

1.3. Suoritusajan arviointi

Tulemme arvioimaan käytettyjen algoritmien suoritusaikoja eli *asymptoottista kertaluokkaa*, mikä on oleellista kun vertailemme, mikä algoritmeista on käytännöllisin - tai ylipäätänsä käytännöllinen missään yhteydessä. Käytämme useille tuttua $\mathcal{O}$-merkintää. Määrittelemme merkinnän reaalifunktioille seuraavasti:

MÄÄRITELMÄ 1.2. Olkoon joukko $A \subset \mathbb{R}$ ylhäältä rajoittamaton ja $f, g : A \to \mathbb{R}_+$ funktioita. Jos on olemassa luvut $C > 0$ ja $x_0 \in A$ siten, että $f(x) \leq C \cdot g(x)$, kun $x \geq x_0$, niin merkitsemme $f = \mathcal{O}(g)$.

Selvemmin ilmaistuna tämä tarkoittaa sitä, että jos funktio g saa suurempia arvoja kuin funktio f, kun niiden syötteet ovat ”suuria”, niin sanomme, että funktio f on tyyppiä $\mathcal{O}(g)$.

ESIMERKKI 1.3. Olkoon $f(x) = \log x$ ja $g(x) = 3x$, niin selvästi $f = \mathcal{O}(g)$. Samoin, jos $h(x) = \frac{1}{3}x^2$, niin sekä $f = \mathcal{O}(h)$ että $g = \mathcal{O}(h)$, sillä h kasvaa nopeasti lineaarisesti kasvavan funktion g ja logaritmisesti kasvavan funktion f ohi.

HUOMAUTUS 1.4. Kannattaa huomata, että esimerkiksi $100 \cdot x = \mathcal{O}(x)$, sillä voimme valita määritelmän luvuksi C lukua 100 suuremman luvun. $\mathcal{O}$-merkinnän sisään

ei ole tapana merkitä kertoimia, sillä ne eivät oleellisesti vaikuta suuruusluokkaan - niin hullulta kun se voikin tuntua. Käytännössä kuitenkin olisi hyvä, jos kertoimet pysyisivät kohtuullisen kokoisina.

Vaikka edellä käsittelimme $\mathcal{O}$-merkinnän määritelmän reaalifunktioille, niin samalla merkintä on tyypillinen myös algoritmien suoritusajan arvioinnissa: useimmiten suoritusajat ovat muotoa $\mathcal{O}(\log n)$, $\mathcal{O}(n^k)$, $\mathcal{O}(k^n)$ tai jokin näiden tulo, missä n on annetun syötteen koko. Järjestelyalgoritmeissa n voi olla esimerkiksi järjestettävien alkioiden määrä. Tässä tutkielmassa käsittelemme kuitenkin yksittäisiä lukuja koskevia algoritmeja, joten usein samalla luvulla n tarkoitamme algoritmille käsiteltäväksi tarkoitettua lukua, esimerkiksi alkulukuehdokasta.

Toivomme luonnollisesti löytävämme algoritmeja, joiden suoritusaika on mahdollisimman pieni. Alkulukutestauksessa arvioimme algoritmien suoritusaikaa eli suoritettavien laskutoimitusten lukumäärää suhteessa syötteen kokoon, eli testattavan luvun n numeroiden lukumäärään nähden. Luvun n b-kantaisessa esityksessä numeroiden lukumäärä on $\lfloor \log_b n \rfloor = \mathcal{O}(\log n)$. Ehdottomasti hyödyllisimpiä ovat algoritmit, joissa saavutamme lopputuloksen *polynomisessa ajassa*, eli suoritusaika on esitettävissä polynomina syötteen koon suhteen. Haluamme siis muodostaa algoritmeja, joiden suoritusaika on polynomi numeroiden lukumäärän $\mathcal{O}(\log n)$ suhteen, siis muotoa $\mathcal{O}(\log^k n)$ jollekin reaaliluvulle $k > 0$. Sen sijaan luvulle n suoritusajaltaan tyyppiä $\mathcal{O}(n)$ tai jopa $\mathcal{O}(n^k)$ olevat algoritmit ovat nopeasti käyttökelvottomia syötteen koon kasvaessa, vaikka käytettävissä olisi kuinka paljon laskentatehoa tahansa.

Seuraava määritelmä auttaa yksinkertaistamaan merkintöjä eräissä tapauksissa:

MÄÄRITELMÄ 1.5. Jos funktio f on tyyppiä $\mathcal{O}(g \log^k g)$ jollekin kiinteälle luvulle k, niin merkitsemme, että $f = \mathcal{O}^{\sim}(g)$.

Edellä asettamamme määritelmän $\mathcal{O}^{\sim}$-merkintä on lähes sama kuin aiempi $\mathcal{O}$-merkintä, mutta sen tarkoituksena on hävittää ylimääräiset logaritmit, jotka eivät käytännössä vaikuta kovin merkittävästi lopulliseen suoritusaikaan: esimerkiksi

$$\mathcal{O}(m \log m \log\log m) = \mathcal{O}^{\sim}(m),$$

mutta tapauksessamme, jossa tutkimme algoritmien kertaluokkaa luvun n numeroiden lukumäärän $\mathcal{O}(\log n)$ suhteen, hävitämme merkinnällä vain sisäkkäiset logaritmit.

1.4. Laskutoimitusten aikavaativuus

Pienillä luvuilla laskeminen tyypillisissä 32- tai 64-bittisissä prosessoreissa on vakioaikaista ja toteutettavissa muutamilla konekielisillä käskyillä, eli nykyisten prosessorien kellotaajuuksilla nanosekunneissa. Vakioaikaiset operaatiot ovat edellä käsitellyillä merkinnöillä tyyppiä $\mathcal{O}(1)$. Kun luvut ylittävät pituudeltaan prosessorin käsittelykyvyn, on lasku pilkottava osiin. Matemaattisessa kirjallisuudessa tietokoneiden laskuoperaatiot usein pilkotaan bitin kokoisiin osiin, bittioperaatioiksi (engl. bit operation), mitä voisi nimittää naiiviksi malliksi tietokoneilla laskemisesta [**4**, s. 7]. Tällöin yhteen- ja vastaavasti vähennyslasku tapahtuvat bitti kerrallaan binäärisen allekkainlaskun tapaisesti.

$$\begin{array}{r} {\scriptstyle 1\quad 11} \\ 1010\,0111 \\ +\,0011\,0110 \\ \hline 1101\,1101 \end{array}$$

Yhteenlaskussa bittioperaatioita siis tapahtuu yhtä monta, kuin on pidemmän summattavan luvun pituus binäärisenä. Koska kokonaisluvun n pituus binäärijärjestelmässä on $\lfloor \log_2 n \rfloor$, niin kahden luvun m ja n yhteenlasku vaatii yhteensä

$$\max\{\log_2 m, \log_2 n\} = \mathcal{O}(\log \max\{m, n\})$$

bittioperaatiota. Naiivissa mallissa myös kertolasku tapahtuu allekkainlaskutyyppisesti vaiheittain useampivaiheisena yhteenlaskuna:

$$\begin{array}{r} 1101 \\ \cdot\quad 0111 \\ \hline 1101 \\ 1\,1010 \\ 11\,0100 \\ +\,000\,0000 \\ \hline 101\,1011 \end{array}$$

Pahimmassa tapauksessa kertolaskussa siis on tehtävä kertojarivin binäärisen pituuden verran yhteenlaskuja - toteutuksesta riippuen nollarivit voidaan laskea tai olla laskematta. Pessimistin periaatteella oletamme, että laskuja tapahtuu mahdollisimman monta ja jokaisessa yhteenlaskussa bittioperaatioita riittää tehdä kerrottavan luvun binäärisen pituuden verran. Helposti saamme kahden kokonaisluvun m ja n kertolaskun bittioperaatioiden lukumääräksi

$$\lfloor \log_2 m \rfloor \cdot \lfloor \log_2 n \rfloor \leq \lfloor \log_2 m \log_2 n \rfloor = \mathcal{O}(\log m \log n).$$

Jakolasku sekä jakojäännöksen selvittäminen ovat kertolaskun kanssa samaa kertaluokkaa, sillä se on toteutettavissa toistettujen vähennyslaskujen avulla.

Kertolasku on toteutettavissa myös esimerkiksi nopeaan Fourier-muunnokseen perustuvilla menetelmillä ajassa $\mathcal{O}^{\sim}(\log n)$=$\mathcal{O}(\log n \log\log n \log\log\log n)$ [**8**, s. 240]. Tämän menetelmän yksityiskohtiin emme pureudu tässä tutkielmassa, mutta kyseisen menetelmän avulla voimme kuitenkin nipistää erityisesti viimeisenä käsittelemämme algoritmin suoritusajasta osan pois.

Todellisuudessa lukuja ei tarvitse laskea aivan bitti kerrallaan, sillä prosessoreissa on tilaa suuremmillekin luvuille. Lisäksi nykyaikaisissa prosessoreissa on olemassa kertolaskupiirejä, jotka pystyvät laskemaan pienten lukujen kertolaskuja vähemmällä vaivalla kuin toistamalla yhteenlaskua. Oleellista tällaisella kertolaskupiirillä on, ettei tulon pituus binäärisenä ylitä prosessorin käsittelykykyä.

1.5. Tarvittavat tulokset

Kokonaislukujen jaollisuus ja kongruenssiyhtälöiden ymmärtäminen ovat kaikkien tulosten perusta jatkossa. Sanomme, että $a \equiv b \mod n$ eli luvut a ja b ovat kongruentteja modulo n, jos $n \mid a-b$ eli n jakaa luvun $a-b$. Kongruenssi modulo n on selvästi ekvivalenssirelaatio. Jos $a \equiv b \mod n$, niin ne kuuluvat samaan jäännösluokkaan $[a]_n = \{x \in \mathbb{Z} \ : \ x \equiv a \mod n\}$. Merkintöjen lyhentämiseksi voimme sanoa, että renkaassa $\mathbb{Z}_n$ luvut a ja b ovat samoja, tai $a = b$. Vaikka usein laskemme jäännösluokilla, emmekä niiden edustajilla, niin käytämme usein merkintää $a = [a]_n$ ja termiä "luku" kuvaamaan alkiota a jäännösluokan sijaan.

Oletamme tunnetuksi paitsi lukuteorian osaamista, myös algebran peruskurssin oleellisimmat tiedot. Algebran perusrakenteet, kuten ryhmät, renkaat ja kunnat ovat myös erityisesti hankalimpia todistuksia ajatellen välttämättömiä esitietoja. Esimerkiksi tarvitsemme tietoa, että rengas $\mathbb{Z}_n$ on kunta jos ja vain jos n on alkuluku [**16**, s. 70]. Tämän seurauksena kaikilla alkuluvun n määräämän kunnan $\mathbb{Z}_n$ alkioilla nollaa lukuun ottamatta on käänteisalkio kertolaskun suhteen.

Käytämme muun muassa seuraavia lukuteoreettisesti merkittäviä tuloksia. Todistuksia emme tässä kappaleessa käsittele, mutta ne ovat löydettävissä lähdemateriaalista.

Yksi klassisimpia lukuteorian tuloksia on kiinalainen jäännöslause:

LAUSE 1.6. *Olkoot luvut* $n_1, ..., n_k \in \mathbb{N}\setminus\{1\}$ *siten, että* $\text{syt}(n_i, n_j) = 1$ *kaikille* $i \neq j$. *Tällöin lineearisella kongruenssiyhtälöryhmällä*

$$\begin{cases} x \equiv b_1 \mod n_1 \\ x \equiv b_2 \mod n_2 \\ \quad\quad \vdots \\ x \equiv b_k \mod n_k \end{cases}$$

on yksikäsitteinen ratkaisu modulo $n := \prod_{i=1}^{k} n_i$.

TODISTUS. Ks. [**4**, s. 51]. □

Lukuteoreettisista funktioista on tarvitsemme jatkossa erityisesti Eulerin φ-funktiota $\varphi : \mathbb{N} \to \mathbb{N}$, $\varphi(n) = \#\{d \leq n \ : \ \text{syt}(d, n) = 1\}$, joka ilmoittaa lukua n pienempien sille *suhteellisten alkulukujen* lukumäärän. Lukuja a ja b sanotaan toisilleen tai keskenään suhteellisiksi alkuluvuiksi, jos $\text{syt}(a, b) = 1$.

Eulerin φ-funktiolle on olemassa useita käytännöllisiä laskusääntöjä:

LAUSE 1.7. *Eulerin* φ*-funktio on multiplikatiivinen, eli jos* $\text{syt}(a, b) = 1$, *niin* $\varphi(ab) = \varphi(a)\varphi(b)$.

TODISTUS. Ks. [**9**, s. 53] □

Alkuluvulle p on selvästi $\varphi(p) = p - 1$. Lisäksi on helppoa huomata, että alkuluvun potenssille p^k on $\varphi(p^k) = p^{k-1}(p-1)$.

Eulerin lauseella on useita hyödyllisiä seurauksia, joista eräitä on Fermat'n pieni lause. Eulerin φ-funktiota tarvitsemme seuraavan lauseen muotoiluun.

LAUSE 1.8 (Eulerin lause). *Jos* $\operatorname{syt}(a, n) = 1$, *niin* $a^{\varphi(n)} \equiv 1 \mod n$.

TODISTUS. Ks. [**9**, s. 63]. Käsittelemme tämän myös myöhemmin. □

Myös algebran perustiedot helpottavat jatkossa eräitä todistuksia. Oletamme tavallisimmin käytetyt algebralliset rakenteet, kuten ryhmät, renkaat ja kunnat tutuiksi. Yksi hyödyllisistä tuloksista on Lagrangen lause, jossa tulee muistaa kertaluvun määritelmä: ryhmän G kertaluku $\operatorname{ord} G$ on sen alkioiden lukumäärä. Yksittäisen alkion $g \in G$ kertaluku $\operatorname{ord} g$ on sen virittämän syklisen aliryhmän kertaluku. Käytämme myös merkintää $\operatorname{ord}_n(a)$, jolla tarkoitamme luokan $[a]_n \in \mathbb{Z}_n$ virittämän aliryhmän kertalukua.

LAUSE 1.9 (Lagrangen lause). *Olkoon G äärellinen ryhmä ja H tämän aliryhmä. Tällöin* $\operatorname{ord} H \mid \operatorname{ord} G$.

TODISTUS. Ks. [**2**, Luku 10, s. 22] □

LUKU 2

Heikot alkulukutestit

Toimivia alkulukutestejä on olemassa monenlaisia, mutta kaikki niistä vain eivät ole riittävän tehokkaita ollakseen käytännöllisiä suurille luvuille. Tutkimme seuraavaksi muutamia yksinkertaisia testejä.

Jaottelemme jatkossa testit *deterministisiin* ja *probabilistisiin*. Deterministiset testit antavat täysin varman vastauksen ja probalistiset antavat nimensä mukaisesti ”todennäköisesti” oikean vastauksen. Hyvässä probabilistisessa testissä pystymme arvioimaan todennäköisyyttä virheelle. Molemmat seuraavista yksinkertaisista alkulukutesteistä ovat deterministisiä, mutta seuraavissa kappaleissa probabilistiset testit osoittautuvat myös hyviksi vaihtoehdoiksi.

2.1. Jakolasku

Ehdottomasti helpoin testi on jo koulusta tuttu jakolasku: jos testattava luku ei ole jaollinen millään itseään pienemmällä luvulla, on sen oltava alkuluku. Itse asiassa jakolasku riittää tehdä huomattavasti vähemmällä määrällä lukuja kaikkien testattavaa lukua pienempien lukujen sijaan: jos testattava luku n ei ole jaollinen millään neliöjuurtaan $\sqrt{n}$ pienemmällä tai yhtä suurella positiivisella kokonaisluvulla, niin se on alkuluku. Todistaminen onnistuu kääntämällä väite: jos n olisi yhdistetty luku, muttei jaollinen millään korkeintaan neliöjuurensa suuruisella luvulla, niin luku n olisi kahden luvun a ja b tulo, joista kumpikin olisi suurempi kuin $\sqrt{n}$. Tällöin siis

$$n = ab > \sqrt{n}\sqrt{n} = n,$$

mikä on ristiriita.

Aluksi toki on syytä selvittää, ettei testattavan luvun n neliöjuuri ole kokonaisluku, muuten jakolaskun tekemisessä ei ole mieltä, sillä tällöin n ei tietenkään voisi olla alkuluku kahden luvun neliönä. Jakolaskuvaiheessa riittäisi itse asiassa jakaa luku n kaikkia neliöjuurtaan pienemmillä alkuluvuilla, mutta näiden tunnistamiseen voisi kulua enemmän aikaa kuin jakolaskuihin. Parillisilla luvuilla jakaminen kuitenkin kaksinkertaistaa laskenta-ajan, joten riittää alussa kokeilla jakaa kahdella ja tämän jälkeen kokeilla ainoastaan parittomilla luvuilla jakamista. Yhdistelemällä edellä olevat päätelmät saamme seuraavan algoritmin:

```
// Jakolaskukeino luvun testaamiseksi alkuluvuksi
static bool onAlkuluku_Jakolasku(BigInteger n)
{
    if (n < 2 || n % 2 == 0) return false;
```

```
    BigInteger sqrt_n_arvio = BigInteger.Pow(2,
       (int)((BigInteger.Log(n, 2))/2 + 1));

    for (BigInteger k = 3; k <= sqrt_n_arvio; k = k + 2)
    {
        if (n % k == 0) return false;
    }
    return true;
}
```

Algoritmissamme heti aluksi tutkimme, onko luku parillinen tai liian pieni ollakseen alkuluku. Tässä alun suuruusluokka- ja parillisuustarkastelu ovat käytännössä hyvin nopeasti tarkastettavissa, sillä parillisuuden selvittämiseksi riittää katsoa viimeisen luvun parillisuutta - binäärimuodossa riittää katsoa viimeistä bittiä.

Toiseksi joudumme laskemaan luvun n neliöjuuren, mikä ei ole itsestäänselvyys suurilla luvuilla: tyypillisesti neliöjuuren laskeminen on jo toteutettu laitteistossa pienille luvuille, mutta suurikokoisille luvuille on neliöjuuren laskeminen myös toteutettava lyhyemmissä pätkissä, mikä onnistuu esimerkiksi babylonialaisella menetelmällä. Voimme kuitenkin oikaista: koska luvun n pituus b-kantaisena oli $\lfloor \log_b n \rfloor$ numeroa, niin sen neliöjuuren pituuden on oltava $\lfloor \log_b \sqrt{n} \rfloor = \lfloor \frac{1}{2} \log_b n \rfloor \leq \frac{1}{2} \lfloor \log_b n \rfloor$ numeroa, eli enintään puolet alkuperäisestä. Voimme siis käyttää varsinaisen neliöjuuren sijaan lukua, jonka pituus on puolet alkuperäisestä ja ylittää neliöjuuren suuruusluokallaan - binäärisenä tämä olisi $\frac{1}{2} \lfloor \log_2 n \rfloor$ kappaletta ykkösiä, eli $2^{\frac{1}{2} \lfloor \log_2 n \rfloor + 1} - 1$. Tämän luvun luominen on helppoudessaan vakioaikainen operaatio, eikä oleellisesti vaikuta laskenta-aikaan.

Neliöjuuren arvion laskemisen jälkeen on laskettava jakojäännös saatua neliöjuuren arviota pienemmillä tai samankokoisilla parittomilla kokonaisluvulla enintään $c \cdot \sqrt{n}/2 = \mathcal{O}(\sqrt{n})$ kertaa, missä vakion c lisäsimme neliöjuuren arvioinnin vuoksi, ja jakojäännöksen laskeminen on jokaisella kierroksella kertaluokaltaan suurimmillaan luokan $\mathcal{O}(\log n \log \sqrt{n}) = \mathcal{O}(\log n \frac{\log n}{2}) = \mathcal{O}(\log^2 n)$ operaatio. Summaamalla yhteen kaikkien vaiheiden asymptoottiset kertaluokat saamme kertaluokaksi

$$\mathcal{O}(1) + \mathcal{O}(\sqrt{n}) \cdot \mathcal{O}(\log^2 n) = \mathcal{O}(\sqrt{n} \log^2 n).$$

Tämä suoritusaika on pienille luvuille varsin toimiva, mutta suurille luvuille se käy nopeasti ylivoimaiseksi: koska luvun n pituus on luokkaa $\mathcal{O}(\log n)$, niin saamamme suoritusaika on eksponentiaalinen luvun n pituuden suhteen.

2.2. Wilsonin lause

Wilsonin lause on nimeänsä kantavaa henkilöä huomattavasti vanhempi lukuteoreettinen tulos, jonka keksimisestä on ollut merkkejä jo nykyisen Irakin alueella eläneen matemaatikon Ibn al Haythamin teksteissä 1000-luvulla. Yleisempi nimi lauseelle tulee englantilaiselta matemaatikolta John Wilsonilta, joka oli väitteen vuonna 1770 esittäneen myös englantilaisen matemaatikon Edward Waringin oppilas. Kumpikaan ei osannut todistaa väitettä ennen Lagrangea, joka todisti tuloksen vuonna 1772 [**6**,

s. 32].

Wilsonin lause on erinomainen esimerkki siitä, miten muotoilultaan varsin yksinkertainen ja ehdoiltaan lupaava lause voi olla käytäntöön sovellettuna täysin hyödytön. Testi on jopa deterministinen, eli antaa täsmällisen tiedon siitä, onko n alkuluku vai ei.

LAUSE 2.1 (Wilsonin lause). *Luku $n > 1$ on alkuluku jos ja vain jos*

$$(n-1)! \equiv -1 \mod n.$$

TODISTUS. Oletamme ensin, että n on alkuluku. Tapauksessa $n = 2$ saamme väitteen muotoon $1 \equiv -1 \mod 2$, mikä on selvää. Riittää siis osoittaa väitteen pitävyys tapauksessa $n \geq 3$. Koska $\mathbb{Z}_n$ on kunta, jos n on alkuluku, niin joukossa $\mathbb{Z}_n$ on jokaiselle nollasta eroavalle alkiolle a kertolaskun suhteen yksikäsitteinen käänteisluku a^{-1} siten, että $aa^{-1} \equiv 1 \mod n$. Luvut, joiden käänteisalkiot ovat ne itse, ratkeavat yhtälöstä $x^2 \equiv 1 \mod n$ ja tällä ratkaisuja on tunnetusti korkeintaan astelukunsa verran eli kaksi. Ratkaisut 1 ja $n-1$ löytyvät helposti. Nyt kertoma

$$(n-1)! = 1 \cdot 2 \cdot \ldots \cdot (n-1)$$

sisältää kaikki kunnan $\mathbb{Z}_n$ alkiot, jotka ovat järjestettävissä alkioita 1 ja $n-1$ lukuun ottamatta käänteisalkiopareiksi, joiden tulo modulo n on 1. Tällöin $(n-1)! = (1 \cdot (n-1)) \cdot 1 \cdot \ldots \cdot 1 \equiv -1 \mod n$.

Jos n on yhdistetty luku, niin sillä on alkulukujakaja p. Jos olisi $(n-1)! \equiv -1 \mod n$, niin myös $(n-1)! \equiv -1 \mod p$, mutta kuitenkin

$$(n-1)! = 1 \cdot 2 \cdot \ldots \cdot p \cdot \ldots \cdot (n-1) \equiv 0 \mod p.$$

□

Wilsonin lause näyttää lupaavalta, sillä se antaa ekvivalentin ehdon sille, että mikä tahansa luku on alkuluku n yhdessä yhtälössä täysin ilman ylimääräisiä muuttujia. Käytännössä testi ei ole kuitenkaan käyttökelpoinen, vaikkei siinä ole edes tarvetta laskea suurilla luvuilla: Kun tulo lasketaan pari kerrallaan ja tuloa vähennetään modulo n, ei käsiään tarvitse liata luvun n neliötä suuremmilla luvuilla. Tästä huolimatta laskutoimituksia on tehtävä niin järkyttävä määrä, että suurien alkulukujen kohdalla tarkistus ei tule koskaan valmiiksi.

Algoritmiksi saamme seuraavaa:

```
// Wilsonin lauseeseen perustuva testi
static bool onAlkuluku_Wilson(BigInteger n)
{
    BigInteger kertoma = 1;
    for (BigInteger indeksi = 1;
            kertoma != 0 && indeksi < n;
            kertoma = (kertoma * indeksi++) % n) ;

    return (kertoma == n - 1);
}
```

Algoritmeja kehittelevä matemaatikko voisi ihmetellä, että miksei kertoman laskemista suoriteta rekursiotyyppisesti, sillä se johtaa usein nopeammin lopputulokseen. Tietokoneessa on valitettavasti rajallinen määrä tilaa suorituspinossa, ja rekursio varaa huomattavan määrän paikkoja pinosta tavalliseen silmukkaan nähden: tässä tapauksessa kertoman laskemiseen olisi laitettava ainakin $n-1$ käskyä peräkkäin, mikä kasvaa nopeasti aivan liian suureksi tietokoneen muistille.

Algoritmissa oleellisin työ tapahtuu kertomaa laskiessa. Pahimmassa tapauksessa suoritusta ei voi lopettaa kesken laskennan kertoman ollessa kongruentti nollan kanssa modulo n, vaan tuloja modulo n on laskettava $n-1$ kappaletta. Näin saamme Wilsonin algoritmin suoritusajaksi

$$(n-1) \cdot (\mathcal{O}(\log n \log n) + \mathcal{O}(\log n^2 \log n)) = \mathcal{O}(n \cdot \log^2 n).$$

Wilsonin lauseen ehdon arvioiminen on siis huomattavasti työläämpää kuin esimerkiksi jakolaskun käyttö, joka oli suoritusajaltaan $\mathcal{O}(\sqrt{n} \log^2 n)$.

LUKU 3

Vahvemmat alkulukutestit

Tässä luvussa käsittelemme aiempaa voimakkaampia ja käytetympiä alkulukutestejä. Ensimmäinen näistä on probabilistinen Millerin ja Rabinin testi, jonka suoritusaika on edellä käsiteltyihin algoritmeihin nähden varsin hyvä; kyseessä onkin yksi käytetyimmistä alkulukutesteistä ja hyvin tyypillinen valinta satunnaisten suurten alkulukujen löytämiseen. Toinen testeistä on Solovayn ja Strassenin testi, joka on myös probabilistinen, mutta vähemmän hyödynnetty kuin Millerin ja Rabinin testi. Lucasin ja Lehmerin testi puolestaan on deterministinen, mutta toimii vain muotoa $2^p - 1$ oleville alkulukuehdokkaille. Suurimmat löydetyt alkuluvut ovat kuitenkin juuri edellä olevaa muotoa.

3.1. Millerin ja Rabinin testi

3.1.1. Fermat'n lause ja Carmichaelin luvut. Kuuluisa ja yksi lukuteorian sekä algebran kurssien merkittävimmistä tuloksista on Fermat'n pieni lause, joka ilmoittaa hyödyllisen ominaisuuden kaikille alkuluvuille. Fermat'n pieni lause seuraa suoraan Eulerin lauseesta:

LAUSE 3.1 (Eulerin lause). *Olkoot n ja a toisilleen suhteellisia alkulukuja ja $n > 1$. Tällöin*

$$a^{\varphi(n)} \equiv 1 \mod n.$$

TODISTUS. Renkaan $\mathbb{Z}_n$ kääntyvien alkioiden, eli yksiköiden, multiplikatiivisen ryhmän $\mathbb{Z}_n^\times$ kertaluku on $\varphi(n)$. Koska ryhmä $\mathbb{Z}_n^\times$ on äärellinen, niin Lagrangen lauseen nojalla tämän minkä tahansa aliryhmän $\{1, a, a^2, ..., a^{k-1}\}$ kertaluku $k = \mathrm{ord}_n(a) \mid \varphi(n)$ eli $\varphi(n) = km$ jollekin jakajalle m. Koska $a^k \equiv 1 \mod n$, niin $a^{\varphi(n)} = a^{km} = (a^k)^m \equiv 1^m \equiv 1 \mod n$. □

LAUSE 3.2 (Fermat'n pieni lause). *Olkoon $p \in \mathbb{P}$ ja $a \in \mathbb{N}$ siten, että $p \nmid a$. Tällöin*

$$a^{p-1} \equiv 1 \mod p.$$

TODISTUS. Väite on Eulerin lauseen erikoistapaus, sillä alkuluvulle p pätee $\varphi(p) = p - 1$ ja kaikille luvun p kanssa suhteellisille alkuluvuille a, eli niille a joille $p \nmid a$, on voimassa $a^{p-1} = a^{\varphi(p)} \equiv 1 \mod p$. □

Alkulukutestauksen kannalta tämä on käännettynä hyvin lupaava tulos: jos kokonaisluvulle n ei päde yhtälö $a^{n-1} \equiv 1 \mod n$, niin n ei voi olla alkuluku. Laskemisen kannalta tulos näyttää kuitenkin vähintäänkin epäilyttävältä: luvun $a^{n-1} \mod n$ laskeminen näyttää vähintäänkin työläältä, erityisesti kun n on "iso".

Tulee tosin huomata, että tässä olemme kiinnostuneita vain siitä, mitä on luvun a^{n-1} jakojäännös jaettaessa luvulla n. Tällöin lukua a^{n-1} ei tarvitse siis laskea loppuun asti, vaan osatuloa $a^k, k < n-1$ voidaan tunnettujen laskusääntöjen mukaisesti vähentää modulo n ikään kuin renkaan $\mathbb{Z}_n$ alkioksi, jolloin käsiteltävien lukujen koko on aina enintään n^2.

Yleisesti käytetty keino laskea yleinen luku $a^k \mod n$ on *toistettu neliöinti* (eng. *repeated squaring*). Siinä laskutoimitusten määrä on logaritminen luvuissa olevien numeroiden määrään nähden: algoritmin tehokkuus perustuu siihen, että lukua a voidaan neliöidä modulo n sen sijaan, että laskettaisiin yksitellen tulo n kappaleesta lukuja modulo n. Jakamalla eksponentti k kahdella saamme

$$a^k = \begin{cases} a(a^2)^{(k-1)/2}, & \text{jos } k \text{ pariton} \\ (a^2)^{k/2}, & \text{jos } k \text{ parillinen.} \end{cases}$$

Tällöin huomaamme, että

$$a^k \mod n = \begin{cases} a(a^2 \mod n)^{(k-1)/2} \mod n, & \text{jos } k \text{ pariton} \\ (a^2 \mod n)^{k/2} \mod n, & \text{jos } k \text{ parillinen,} \end{cases}$$

jolloin voimme muodostaa rekursiivisen algoritmin lopullisen luvun laskemiseksi, kun hajautamme sisäkkäiset potenssilaskut edelleen vastaavasti neliöiksi.

```
// Rekursiivinen toteutus toistetulle neliöinnille.
// Laskee luvun a^k mod n.
static BigInteger PotenssiMod(BigInteger a, BigInteger k,
   BigInteger n)
{
    if (k == 0) return 1;
    else if (k == 1) return a;
    else if (k.IsEven) return PotenssiMod((a * a) % n,
        k / 2, n) % n;
    else
        return a * PotenssiMod((a * a) % n, (k-1) / 2, n) % n;
}
```

Algoritmissa laskenta-aika on pisimmillään, kun eksponentin binääriesitys sisältää pelkkiä ykkösiä, eli kun luku k on muotoa $2^{l+1} - 1$: Tällöin laskutoimitusten määrä on maksimaalinen, sillä eksponentti on pariton aina jaettaessa se kahdella, jolloin kertolaskuja on tehtävä yksi enemmän.

Laskutoimituksia kertyy siis jokaisella kierroksella yksi $\mathcal{O}(\log^2 n)$-kertolasku, $\mathcal{O}(\log k)$-jakolasku, $\mathcal{O}(\log^2 n)$-jakojäännöslasku ja $\mathcal{O}(\log k)$-vähennyslasku. Kierroksia kertyy $l = 4\log_2 k$ kappaletta, mikä tekee algoritmista asymptoottiselta kertaluokaltaan

$$\begin{aligned} \mathcal{O}(\log k) \cdot (\mathcal{O}(\log^2 n) + \mathcal{O}(\log k) + \mathcal{O}(\log^2 n) + \mathcal{O}(\log k)) &= \mathcal{O}(\log k \log^2 n + \log^2 k) \\ &= \mathcal{O}(\log^3 \max(k, n)). \end{aligned}$$

Fermat'n lauseen ehto testattavalle luvulle n on siis tutkittavissa logaritmisessa ajassa ja ehdon tarkistamista eri kantaluvuilla a kutsumme Fermat'n alkulukutestiksi, joka on suoritusajaltaan käytännössä sama kuin toistettu neliöinti eli tässä

tapauksessa $\mathcal{O}(k \log^3 n)$, missä k on testattujen kantalukujen a lukumäärä. Tehokkaammilla kerto- ja jakolaskualgoritmeilla on helppoa huomata, että voimme korvata $\mathcal{O}(\log^2 n)$-laskut $\mathcal{O}^{\sim}(\log n)$-luokan laskutoimituksilla, jolloin kokonaissuoritusaika on $\mathcal{O}^{\sim}(k \log^2 n)$. Nopeasti päättelemällä voimme kuitenkin huomata, että Fermat'n testi ei kuitenkaan varsinaisesti kerro, onko testattava luku alkuluku, vaan se *voi* paljastaa yhdistetyn luvun, muttei aina.

On itse asiassa olemassa *Carmichaelin lukuja*, jotka toteuttavat Fermat'n alkulukutestin ehdon jokaisella kantaluvulla a, paitsi tietenkin niillä, joille $\text{syt}(a, n) > 1$ kun n on testattava luku [**4**, s. 130]. Esimerkiksi $561 = 3 \cdot 11 \cdot 17$ on tälläinen luku: $a^{560} \equiv 1 \mod 561$ aina, kun a ei ole jaollinen luvuilla $3, 11$ tai 17. Tämä romuttaa testin todennäköisyysmielessä: lukua a valittaessa satunnaisesti on Carmichaelin luvun alkutekijään osuminen melko epätodennäköistä kaikkien muiden valintojen "valehdellessa" testattavan luvun olevan alkuluku. Määrittelemme selkeyden vuoksi Carmichaelin luvut seuraavasti:

MÄÄRITELMÄ 3.3. Yhdistetty luku n on Carmichaelin luku, jos jokaiselle luvun n kanssa suhteelliselle alkuluvulle a on voimassa

$$a^{n-1} \equiv 1 \mod n.$$

Tarvitsemme jatkossa melko usein myös primitiivisen juuren määritelmää:

MÄÄRITELMÄ 3.4. Jos on olemassa $a \in \mathbb{Z}_n^{\times}$, joka virittää koko ryhmän $a \in \mathbb{Z}_n^{\times}$, niin kutsumme lukua a primitiiviseksi juureksi modulo n.

Primitiivista juurta ei ole olemassa kuin harvoissa tapauksissa. Seuraava lause antaa rajoitteen luvulle n.

LAUSE 3.5. *On olemassa primitiivinen juuri modulo n täsmälleen silloin, kun*

$$n = 2, 4, p^k \text{ tai } 2p^k$$

jollekin parittomalle alkuluvulle p ja kokonaisluvulle $k \geq 1$.

TODISTUS. Ks. [**10**, ss. 186-191]. □

Carmichaelin luvuille on todistettu muun muassa seuraava tulos, jota tulemme hyödyntämään Solovayn ja Strassenin testin toimivuuden todistamisessa:

LAUSE 3.6. *Jos luku n on Carmichaelin luku, niin n on pariton eikä jaollinen minkään luvun neliöllä. Lisäksi kaikille luvun n alkulukutekijöille p pätee $p-1 \mid n-1$.*

TODISTUS. Olkoon n Carmichaelin luku, jolloin n on yhdistetty ja kaikille $a = 1, ..., n-1$, jotka ovat suhteellisia alkulukuja luvulle n, pätee $a^{n-1} \equiv 1 \mod n$. Jos $n \geq 2$, niin $\text{syt}(n-1, n) = 1$ ja siten

$$1 \equiv (n-1)^{n-1} \equiv (-1)^{n-1} \equiv -1 \mod n,$$

mikä on ristiriita.

Todistaaksemme neliöttömyyden merkitsemme $n = p^2m$ ja $a = 1 + pm$. Tällöin

$$\begin{aligned} a^p &\equiv (1+pm)^p \\ &\equiv \tbinom{p}{0}1^p(pm)^0 + \tbinom{p}{1}1^{p-1}(pm)^1 + ... + \tbinom{p}{p}1^0(pm)^p \\ &= 1 + p^2m + \tbinom{p}{2}p^2m^2 + ... + p^pm^p \\ &\equiv 1 \mod n, \end{aligned}$$

ja koska $a \neq 1$ ja p on alkuluku, niin luvun a kertaluku on p. Koska n on Carmichaelin luku, niin $a^{n-1} \equiv 1 \mod n$ ja tällöin luvun a kertaluvun p pitäisi jakaa $n-1$, mutta myös $p \mid n$, mikä tuottaa ristiriidan, sillä muuten pitäisi olla $p = 1$.

Nyt siis Carmichaelin luku n on paitsi pariton, niin myös neliötön. Riittää enää osoittaa, että kaikille luvun $n =: p_1 \cdots p_k$ alkulukutekijöille p_i pätee $p_i - 1 \mid n - 1$. Lauseen 3.5 mukaan kaikissa ryhmissä $\mathbb{Z}_{p_i}^\times$ on olemassa primitiivinen juuri a_i. Erityisesti $\mathrm{syt}(a_i, p_i) = 1$ kaikille $1 \leq i \leq k$. Tällöin kiinalaisen jäännöslauseen nojalla on olemassa sellainen $a \in \mathbb{Z}_\mathrm{n}$, jolle $a \equiv a_i \mod p_i$ kaikille $1 \leq i \leq k$. Tällöin $\mathrm{syt}(a, n) = 1$ ja koska n on Carmichaelin luku, niin $a^{n-1} \equiv 1 \mod n$. Koska $p_i \mid n$, niin $a^{n-1} \equiv 1 \mod p_i$ kaikille $1 \leq i \leq k$. Nyt a on nyt primitiivinen juuri kaikissa ryhmissä $\mathbb{Z}_{p_i}^\times$, joten $\mathrm{ord}_{p_i}(a) = p_i - 1$ ja siten $p_i - 1 \mid n - 1$.

□

Saamme välittömästi edellisen lauseen avulla johdettua lisäominaisuuden Carmichaelin luvuille:

Seuraus 3.7. *Jokainen Carmichaelin luku on vähintään kolmen eri parittoman alkuluvun tulo.*

Todistus. Edellisen lauseen nojalla riittää osoittaa, ettei Carmichaelin luku n voi olla tasan kahden erin alkuluvun tulo. Oletamme vastoin väitettä, että $n = pq$ ja symmetrian vuoksi voimme olettaa, että $p < q$. Lauseen 3.6 nojalla $q - 1 \mid n - 1 = pq - 1 = p(q-1) + p - 1$, jolloin luvun $q-1$ täytyy jakaa luku $p-1$, mikä on oletuksen $p < q$ kannalta mahdotonta. □

Carmichaelin luvut ovat suhteellisen harvassa, mutta onneksi tämä puute Fermat'n alkulukutestissä on paikattavissa.

3.1.2. Millerin Rabinin alkulukutesti. Seuraavaksi tutkimme Fermat'n alkulukutestin erästä vahvennusta, jossa ei ole varsinaisia heikkouksia, kuten Carmichaelin luvut. Muistamalla, että Fermat'n lauseen mukaan alkuluvulle n pätee yhtälö

$$a^{n-1} \equiv 1 \mod n$$

kun $n \nmid a$. Kun $n \geq 3$, niin tässä yhtälössä $n - 1$ on parillinen luku. Voimme siis kirjoittaa luvun $n - 1 = 2^s \cdot d$, missä $s = \max\{k \in \mathbb{N} : 2^k | n - 1\}$, jolloin luku d on pariton. Tällöin suoraan laskemalla saamme luvuille a, joille $n \nmid a$ yhtälön

$$a^{n-1} - 1 = (a^d)^{2^s} - 1 = (a^d - 1)\prod_{i=0}^{s-1}(a^{2^i \cdot d} + 1) \equiv 0 \mod n$$

voimaan. Koska $\mathbb{Z}_\mathrm{n}$ on kunta, ei siellä ole nollan jakajia - toisin sanoen kahden tai useamman luvun tulo ei voi olla nolla, ellei jokin tulon tekijöistä ole nolla. Nyt siis

alkuluku n jakaa joko luvun $a^d - 1$ tai jonkin luvuista $a^{2^r \cdot d} + 1$, $r = 0, 1, \ldots, s-1$. Tällöin voimme siis päätellä, että

$$a^d \equiv 1 \text{ tai } a^{2^r \cdot d} \equiv -1 \mod n \text{ kun } n \nmid a$$

jollekin $r = 0, 1, \ldots, s-1$.

Olemme siis onnistuneet johtamaan Fermat'n lauseelle seurauksen:

SEURAUS 3.8 (Millerin ja Rabinin ehto). *Olkoon luku $a \in \mathbb{N}$ siten, että $p \nmid a$. Jos p on alkuluku ja luku $s = \max\{k \in \mathbb{N} : 2^k \mid p-1\}$ siten, että $p - 1 = 2^s \cdot d$, missä d on pariton, niin on voimassa joko*

$$a^d \equiv 1 \mod p \tag{3.1}$$

tai

$$a^{2^r \cdot d} \equiv -1 \mod p \text{ jollekin } r = 0, 1, \ldots, s-1. \tag{3.2}$$

Kääntäen Millerin ja Rabinin ehdon saamme aikaiseksi *Millerin ja Rabinin alkulukutestin*: jos testattava luku n ei toteuta kumpaakaan yhtälöistä (3.1) tai (3.2) jollakin kantaluvulla a, niin luku n ei voi olla alkuluku. Selvästi Millerin ja Rabinin ehtojen laskemiseen kuluu suunnilleen yhtä paljon aikaa kuin Fermat'n lauseen ehdon tarkastamiseen, mutta ohjelmoituna tämä vaatii useamman ehtolauseen tarkistamista jokaisella kierroksella.

Miksi Millerin ja Rabinin testi olisi kuitenkaan oleellisesti parempi kuin Fermat'n alkulukutesti? Vastauksena tähän kysymykseen ovat jo aiemmin mainitut Carmichaelin luvut, sillä osoittautuu, että Millerin ja Rabinin testillä ei vastaavaa heikkoutta ole.

LAUSE 3.9. *Jos $n > 4$ on pariton yhdistetty luku, niin Millerin ja Rabinin testin ehdoista* (3.1) *ja* (3.2) *jompikumpi on voimassa enintään yhdelle neljäsosalle kantaluvuista $a \in \{1, 2, \ldots, n-1\}$.*

Tämän tuloksen todistus on melko vaativa, mutta tulos itse on erittäin hyödyllinen. Todistaaksemme sen teemme alkuun muutaman määritelmän ja todistamme joitakin lemmoja.

MÄÄRITELMÄ 3.10. Sanomme, että luku $b \in \{1, 2, ..., n-1\}$ on *todistaja* (engl. *witness*) luvulle n, jos

(1) $b^{n-1} \not\equiv 1 \mod n$ tai
(2) on olemassa $i \in \{0, 1, ..., s\}$ siten, että $2^i \mid n-1$ ja $1 < \text{syt}(b^{\frac{n-1}{2^i}} - 1, n) < n$.

Muodostamme aluksi kaksi joukkoa: olkoon ensimmäinen niistä Rabinin omien merkintöjen mukaisesti todistajien joukko

$$W_n = \{1 \leq b < n : b \text{ on todistaja luvulle } n\}.$$

Olkoot $s = \max\{i \in \mathbb{N} \ : \ 2^i \mid n-1\}$ ja $d = \frac{n-1}{s}$ kuten edellä. Selvästi, jos luku n on todistajien joukossa W_n, niin se on yhdistetty: jos ehto $b^n \not\equiv 1$ ei toteudu, niin luku n ei voi Fermat'n lauseen nojalla olla alkuluku, ja jos lukujen $b^{(n-1)/2^i} - 1$ ja n suurin yhteinen tekijä on suurempi kuin yksi, niin luvulla n on olemassa itseään

pienempi ja ykköstä suurempi jakaja.

Todistajien joukon W_n lisäksi olkoon toinen joukko $E_n = \{1 \leq a \leq n-1 : \text{syt}(a,n) = 1\}$ luvulle n sitä pienempien suhteellisten alkulukujen joukko. Tällöin siis Eulerin φ-funktion arvo luvulle n on $\varphi(n) = \#E_n$.

Pääsemme lauseen 3.9 todistukseen käsiksi todistamalla seuraavan väitteen.

LAUSE 3.11. *Parittomalle yhdistetylle luvulle $n > 4$ on voimassa $\#W_n \geq \frac{3}{4}(n-1)$.*

Pisin osuus 1/4-lauseen todistamista on juuri edellisen lauseen todistaminen, mikä vaatii useita lemmoja. Merkintöjen lyhentämiseksi määrittelemme kokonaislukuvektorille $m = (m_1, ..., m_k)$ jakojäännösvektorin $b \mod m = (s_1, ..., s_k)$, missä $s_i = b \mod m_i, 0 \leq s_i < m_i$ ja $m_i \geq 2$ kaikille $i = 1, ..., k$.

LEMMA 3.12. *Olkoot luvut m_i siten, että $m_i \mid n$ kaikilla $i = 1, ..., k$ ja $\text{syt}(m_i, m_j) = 1$ kaikilla $i \neq j$. Merkitään $m = (m_1, ..., m_k)$. Tällöin jokaiselle jakojäännösvektorille $s = (s_1, ..., s_k), s_1 \in E_{m_1}, ..., s_k \in E_{m_k}$ joukoissa*

$$E_n \cap \{b : b \bmod m = s\}$$

on sama määrä alkioita.

TODISTUS. Olkoon luku $P = \prod_{\substack{p \in \mathbb{P}, p \mid n, p \nmid m_i \\ 1 \leq i \leq k}} p$ niiden alkulukujen tulo, jotka jakavat luvun n, mutta eivät yhtäkään luvuista m_i. Jos tällaisia alkulukuja p ei ole, niin määrittelemme, että $P = 1$.

Oletamme ensin, että $P \geq 2$. Kiinalaisen jäännöslauseen nojalla on olemassa kokonaisluku b siten, että $b \mod (m, P) = (s_1, ..., s_k, 1)$, sillä luvut m_i ja P ovat suhteellisia alkulukuja keskenään. Koska lukujen m_i ja P tulo on enintään n, niin voimme olettaa, että $1 \leq b < n$.

Nyt $b \in E_n$, sillä muutoin jokin alkuluku p jakaisi luvut b ja n. Tällöin olisi voimassa joko $p \mid m_i$ jollain $i = 1, ..., k$ tai $p \mid P$. Koska jakojäännösvektorin luvuille s_i pätee kaikilla $i = 1, ..., k$

$$s_i = b - q_i m_i,$$

niin jos alkuluku p jakaisi jonkin luvuista m_i, niin se jakaisi myös luvun s_i, mikä on ristiriidassa oletuksen $\text{syt}(s_i, m_i) = 1$ kanssa. Jos taas sama p jakaisi luvun P, niin vastaavasti päätyisimme ristiriitaan $p \mid 1$.

Määrittelemme funktion

$$f : E_n \to E_{m_1} \times ... \times E_{m_k}, f(a) = (a \bmod m_1, ..., a \bmod m_k).$$

Lähtöjoukko E_n on renkaan $\mathbb{Z}_n$ yksiköistä koostuvana joukkona ryhmä. Koska jokainen $E_{m_i}, 1 \leq i \leq k$ on ryhmä, niin myös maalijoukko $E_{m_1} \times ... \times E_{m_k}$ on ryhmä. Funktio f on hyvin määritelty, sillä mielivaltaiselle luvulle $a \in E_n$ pätee $f(a) = (a \mod m_1, ..., a \mod m_k)$. Tässä luku a on suhteellinen alkuluku luvulle n ja luvut m_i ovat luvun n jakajia, joten luvun a on oltava suhteellinen alkuluku kaikille m_i, $i = 1, ..., k$. Tällöin siis $a \mod m_i \in E_{m_i}$ kaikille $i = 1, ..., k$ ja siten

$f(E_n) \subset E_{m_1} \times ... \times E_{m_k}$.

Funktion f ominaisuuksien nojalla voimme huomata sen olevan homomorfismi: kaikille luvuille $a, b \in E_n$ pätee

$$\begin{aligned} f(ab) &= (ab \mod m_1, ..., ab \mod m_k) \\ &= (a \mod m_1 \cdot b \mod m_1, ..., a \mod m_k \cdot b \mod m_k) \\ &= (a \mod m_1, ..., a \mod m_k)(b \mod m_1, ..., b \mod m_k) \\ &= f(a)f(b), \end{aligned}$$

Funktio f on myös surjektio: jos vektori $s = (s_1, ..., s_k) \in E_{m_1} \times ... \times E_{m_k}$, niin kiinalaisen jäännöslauseen nojalla on olemassa luku b_s siten, että $b_s \mod m_i = s_i$ kaikille $i = 1, ..., k$. Voimme olettaa, että $b_s < M$, ja koska $M \leq n$, on nyt $b_s < n$.

Jos $P = 1$, niin voimme vastaavasti kuin edellä valita kiinalaisen jäännöslauseen nojalla luvun $b < m_1 m_2 \cdot ... \cdot m_k \leq n$, jolle $b \mod m = s$. Tässäkin tapauksessa $b \in E_n$: Jos olisi alkuluku $p \mid b, n$, niin myös $p \mid m_i$ jollekin $i = 1, ..., k$, sillä $P = 1$. Tällöin $s_i = b - q_i m_i$, jolloin $p \mid s_i$ ja $s_i \notin E_n$, mikä on ristiriita oletuksen kanssa.

Nyt siis jokaiselle maalijoukon vektorille $s \in E_{m_1} \times ... \times E_{m_k}$ on olemassa luku $b_s \in E_n$ siten, että $f(b_s) = s$, joten $f(E_n) = E_{m_1} \times ... \times E_{m_k}$.

Nyt $E_n \cap \{b : b \mod m = s\} = \{b \in E_n : f(b) = s\} = f^{-1}(s)$ ja funktion f surjektiivisuuden nojalla alkukuvat $f^{-1}(s)$ ovat epätyhjiä kaikille $s \in E_{m_1} \times ... \times E_{m_k}$.

Alkuperäinen väite on siis yhtäpitävää sen kanssa, että jokaisessa alkukuvassa $f^{-1}(s)$ on yhtä monta alkiota kaikilla $s \in E_{m_1} \times ... \times E_{m_k}$. Olkoon s kiinnitetty, jolloin voimme valita alkion $b_s \in f^{-1}(s)$. Funktion f homomorfisuuden nojalla

$$f^{-1}(s) = \{b_s \cdot k \ : \ k \in \mathrm{Ker}(f)\},$$

jolloin supistussäännön nojalla joukon $f^{-1}(s)$ alkioiden lukumäärä on täsmälleen sama kuin ytimen $\mathrm{Ker}(f)$, eikä se riipu vektorista s.

□

Seuraavaksi todistamme tuloksia, jotka hieman auttavat rajaamaan todistajien joukkoa W_n:

LEMMA 3.13. *Olkoot $p > 2$ alkuluku, $n = p^k, k \geq 2$, $s = \mathrm{syt}(\varphi(p^k), n-1)$ ja $m = \frac{\varphi(p^k)}{s}$. Tällöin enintään $\frac{\varphi(n)}{m}$ kappaletta luvuista $b \in E_n$ ei ole joukossa W_n.*

TODISTUS. Jos jokin luku $b \in E_n$ ei ole todistaja eli $b \notin W_n$, niin luku b ei toteuta kumpaakaan määritelmän 3.10 ehdoista (1) eikä (2). Riittää siis osoittaa, että ehto (1) eli $b^{n-1} \equiv 1 \mod n$ toteutuu korkeintaan $\frac{\varphi(n)}{m}$ kappaleelle lukuja b. Lauseen 3.5 nojalla renkaassa $\mathbb{Z}_{p^k}$ on olemassa primitiivinen juuri, olkoon se a. Primitiivisen juuren määritelmän nojalla a virittää renkaan $\mathbb{Z}_{p^k}$ kertolaskun suhteen kääntyvien alkioiden ryhmän $\mathbb{Z}_{p^k}^\times$, jossa on $\varphi(p^k)$ alkiota. Tällöin $a^t \equiv 1 \mod p^k$ jos ja vain jos $\varphi(p^k) \mid t$: jos $a^t \equiv 1 \mod p^k$, mutta $\varphi(p^k) \nmid t$, niin $t = d \cdot \varphi(p^k) + r, 0 < r < \varphi(p^k)$, jolloin

$$1 \equiv a^t \equiv a^{d \cdot \varphi(p^k) + r} \equiv (a^{\varphi(p^k)})^d \cdot a^r \underset{(1)}{\equiv} 1^d \cdot a^r \equiv a^r \mod p^k,$$

missä yhtälö (1) seuraa Eulerin lauseesta. Eulerin lause on käytettävissä, sillä luvut a ja p^k ovat toisilleen suhteellisia alkulukuja; muutoin a ei voisi olla primitiivinen juuri. Tällöin luvun a kertaluku $\operatorname{ord} a \leq r < \varphi(p^k)$, vaikka $\operatorname{ord} a = \varphi(p^k)$. Väitteen toinen puoli seuraa suoraan Eulerin lauseesta.

Olkoon $b \in E_n$ mielivaltainen, jolloin luvut b ja p^k ovat keskenään suhteellisia alkulukuja, sillä $n = p^k$. Nyt $b \equiv a^r \mod p^k$ aiemmin valitulle primitiiviselle juurelle a ja sen eksponentille $0 \leq r < \varphi(p^k)$.

Oletusten nojalla n on yhdistetty luku. Jos $b^{n-1} \equiv 1 \mod n$, niin koska $b^{n-1} \equiv a^{r(n-1)} \mod p^k$, on voimassa yhtälö

$$a^{r(n-1)} \equiv 1 \mod p^k.$$

Tällöin $\varphi(p^k) \mid r(n-1)$ luvun a ollessa primitiivinen juuri modulo p^k. Luvun m määritelmän nojalla $dsm = d\varphi(p^k) = r(n-1)$ jollekin jakajalle d. Koska luvut m ja $n-1$ ovat oletuksen $\operatorname{syt}(\varphi(p^k), n-1) = s$ nojalla keskenään suhteellisia alkulukuja, pätee $m \mid r$, jolloin $r = hm$ jollekin jakajalle h.

Koska $0 \leq r < \varphi(p^k)$, niin $0 \leq h < \frac{\varphi(p^k)}{m}$. Luku $r = hm$ voi siis saada enintään $\frac{\varphi(p^k)}{m}$ eri arvoa, mikä koskee myös lukua a^r, joka puolestaan on luvun b jakojäännös modulo p^k. Jokaista jakojäännöstä vastaa lemman 3.12 nojalla sama määrä lukuja b ja tämä lukumäärä on $\frac{\varphi(n)}{\varphi(p^k)}$.

Ehto $b^{n-1} \equiv 1 \mod n$ voi olla siis voimassa korkeintaan $\frac{\varphi(p^k)}{m}$ jakojäännökselle, ja jokaiselle näistä jakojäännöksistä on olemassa $\frac{\varphi(n)}{\varphi(p^k)}$ kappaletta lukuja b, jotka toteuttavat ehdon. Kertolaskulla saamme siis lukujen b lukumäärälle ylärajaksi

$$\frac{\varphi(p^k)}{m} \cdot \frac{\varphi(n)}{\varphi(p^k)} = \frac{\varphi(n)}{m}.$$

□

Seuraava lemma on edellisestä yleisempi tapaus:

LEMMA 3.14. *Olkoon p_1 ja p_2 keskenään erisuuret parittomat alkuluvut, $q_1 = p_1^{k_1}, q_2 = p_2^{k_2}$, missä $k_1, k_2 \geq 1$, ja luku n siten, että $q_1 q_2 \mid n$. Olkoon $s_i = \operatorname{syt}(\varphi(q_i), n-1)$ ja $m_i = \frac{\varphi(q_i)}{s_i}$, kun $i = 1, 2$. Tällöin enintään $\frac{\varphi(n)}{m_1 m_2}$ kappaletta luvuista $b \in E_n$ ei ole joukossa W_n. Lisäksi, jos ainakin toinen luvuista $s_i, i = 1, 2$ on parillinen, niin enintään $\frac{\varphi(n)}{2m_1 m_2}$ kappaletta luvuista $b \in E_n$ ei ole joukossa W_n.*

TODISTUS. Ensimmmäisen väitteen todistus on lähes samanlainen kuin lemmalle 3.13: Voimme valita primitiiviset juuret a_1, a_2 alkulukupotensseille q_1, q_2 kuten aiemmin. Mielivaltaiselle luvulle $b \in E_n$ on olemassa eksponentit $0 \leq r_i < \varphi(q_i)$ siten, että $a_i^{r_i} \equiv b \mod q_i$, $i = 1, 2$.

Jos nyt $b^{n-1} \equiv 1 \mod n$, niin $b^{n-1} \equiv a_i^{r_i(n-1)} \equiv 1 \mod q_i$, jolloin $d_i s_i m_i = d_i \varphi(q_i) = r_i(n-1)$ joillekin jakajille d_i kun $i = 1, 2$. Tällöin luvut m_i jakavat luvut r_i lukujen m_i ja $n-1$ ollessa suhteellisia alkulukuja, jolloin on olemassa luvut h_i

siten, että $r_i = h_i m_i$. Vastaavin perustein kuin aiemmin luvut r_i voivat saada $\frac{\varphi(q_i)}{m_i}$ eri arvoa, jolloin myös luvut $a_i^{r_i}$ voivat saada saman verran eri arvoja, jotka puolestaan ovat luvun b jakojäännöksiä modulo q_i. Lauseella 1.7 toteamamme Eulerin φ-funktion multiplikatiivisuuden nojalla eri jakojäännösvektoreita on nyt enintään

$$\frac{\varphi(q_1)}{m_1} \cdot \frac{\varphi(q_2)}{m_2} = \frac{\varphi(q_1 q_2)}{m_1 m_2}$$

kappaletta. Jokaista jakojäännöstä vastaa lemman 3.12 nojalla sama määrä lukuja b ja tämä lukumäärä on $\frac{\varphi(n)}{\varphi(q_1 q_2)}$.

Ehto $b^{n-1} \equiv 1 \mod n$ voi olla siis voimassa korkeintaan $\frac{\varphi(q_1 q_2)}{m_1 m_2}$ jakojäännökselle, ja jokaiselle näistä jakojäännöksistä on olemassa $\frac{\varphi(n)}{\varphi(q_1 q_2)}$ kappaletta lukuja b, jotka toteuttavat ehdon. Kertolaskulla saamme siis lukujen b lukumäärälle ylärajaksi

$$\frac{\varphi(q_1 q_2)}{m_1 m_2} \cdot \frac{\varphi(n)}{\varphi(q_1 q_2)} = \frac{\varphi(n)}{m_1 m_2}.$$

Jälkimmäisen väitteen todistaaksemme oletamme, että ainakin toinen luvista $s_j = \text{syt}(\varphi(q_j), n-1), j = 1, 2$ on parillinen. Olkoon funktio $e : \mathbb{N} \to \mathbb{N} \cup \{0\}, e(n) = \max\{k : 2^k \mid n\}$. Tällöin tilanteemme jakautuu kahteen tapaukseen:

$$e(s_1) = e(s_2) \text{ ja } e(s_1) \neq e(s_2).$$

Olkoon nyt $e_1 := e(s_1)$ ja $e_2 := e(s_2)$. Ensimmäisessä tapauksessa $e_1 = e_2 =: e$. Nyt siis $s_j = 2^e c_j, j = 1, 2$, missä luvut c_j ovat parittomia ja $e \geq 1$, sillä oletimme ainakin toisen luvuista s_j olevan parillinen. Koska $s_j \mid n-1$, niin $n-1 = 2^f n'$ jollekin parittomalle luvulle n' ja eksponentille $f \geq e$. Nyt valitsemme luvun d siten, että $d = 2^{e-1} n'$, jolloin $s_1, s_2 \nmid d$, mutta $s_1, s_2 \mid 2d$: näin pätee, sillä luvun d valitsimme siten, että $2^{f+e-1} d = n-1$, ja selvästi $2^e \nmid 2^{e-1}$ sekä luku n' on pariton, jolloin $2^e \nmid d$.

Olkoon $b \in E_n$ sellainen, että $b^{n-1} \equiv 1 \mod n$. Olkoon myös luvut $a_j, r_j, j = 1, 2$ kuten aiemmin. Tällöin $b \equiv a_j^{r_j(n-1)} \mod q_j, j = 1, 2$. Nyt on jälleen olemassa luvut h_1, h_2 siten, että $r_j = h_j m_j$. Jos h_1 on parillinen ja h_2 pariton pätee kaksi yhtälöä:

$$\varphi(q_1) \nmid h_1 m_1 d(i) \tag{3.3}$$

ja

$$\varphi(q_2) \mid h_2 m_2 d. \tag{3.4}$$

Ensimmäisen väitteen saamme osoitettua helposti todeksi osoittamalla vastaväitteen vääräksi: jos $\varphi(q_1) \mid h_1 m_1 d$, niin $s_1 m_1 \mid h_1 m_1 2^{e-1} n'$ ja erityisesti $s_1 \mid h_1 2^{e-1} n'$, jolloin h_1 olisi oltava parillinen, mutta tämä on oletuksemme kanssa ristiriidassa.

Koska $\varphi(q_2) = s_2 m_2$, niin väitteen (3.4) kohdalla riittää osoittaa, että $s_2 \mid h_2 2^{e-1} n'$, eli $2^e c_2 \mid h_2 2^{e-1} n'$. Luvun h_2 parillisuuden myötä riittää enää siis huomata, että $c_2 \mid n'$, sillä $s_2 = 2^e c_2 \mid 2^f n' = (n-1)$ ja $e \leq f$ sekä c_2 on pariton, jolloin väite on selvä.

Nyt sitten väitteiden (3.3) ja (3.4) nojalla

$$b^d \not\equiv 1 \mod q_1 \text{ ja } b^d \equiv 1 \mod q_2,$$

mikä tarkoittaa sitä, että $1 < \text{syt}(b^d - 1, n) < n$, jolloin luku $d = 2^{e-1}n' = \frac{n-1}{2^i}$ jollekin i näyttää luvun $b \in E_n$ olevan todistaja, eli joukossa W_n. Täysin vastaavaan tilanteeseen päädymme, jos h_2 onkin pariton ja h_1 parillinen. Lukujen h_1 ja h_2 parillisuustilanteet ovat kaikki yhtä yleisiä, ja koska näitä parillisuustilanteita on neljä, joista kahdessa $b \in W_n$, niin ainakin puolet luvuista $b \in E_n$, joille $b^{n-1} \equiv 1 \mod n$, ovat todistajia. Nyt siis tapauksessa $e_1 = e_2$ enintään $\frac{1}{2} \cdot \frac{\varphi(n)}{m_1 m_2} = \frac{\varphi(n)}{2m_1 m_2}$ luvuista $b \in E_n$ eivät ole todistajia.

Riittää enää näyttää väitteen osuvuus tapauksessa, jossa $e_1 \neq e_2$. Voimme muuttaa tämän tapauksen muotoon $e_1 > e_2$: Jos s_1 on parillinen ja s_2 pariton, niin selvästi $e_1 \geq 1$ ja $e_2 = 0$. Tekemättä oletuksia siitä, kumpi luvuista s_1 ja s_2 ainakin on parillinen selviämme vähemmällä työllä. Jälleen $s_j = 2^{e_j} c_j, j = 1, 2$ ja $n - 1 = 2^f n'$ sekä $f \geq e_1 > e_2$. Nyt valitsemme luvun $d = 2^{e_2} n'$, jolloin $s_2 \mid d$, mutta $s_1 \nmid d$, $s_1 \nmid 2^{e_1 - e_2 - 1} d$ ja $s_1 \mid 2^{e_1 - e_2} d$. Olkoon taas luku $b \in E_n$ siten, että $b^{n-1} \equiv 1 \mod n$ ja sitä vastaavat luvut a_1, a_2, r_1 ja r_2 siten, että $b \equiv a_j^{r_j} \equiv 1 \mod n$. Olkoon myös luvut h_1, h_2 siten, että $r_j = h_j m_j, j = 1, 2$. Nyt ovat voimassa ehdot

$$\varphi(q_1) \mid h_1 m_1 d \text{ täsmälleen silloin, kun } 2^{e_2 - e_1} \mid h_1 \tag{3.5}$$

ja

$$\varphi(q_2) \mid h_2 m_2 d. \tag{3.6}$$

Ensimmäisen näistä voimme todistaa helposti: Koska $\varphi(q_1) = s_1 m_1$, niin ehto (3.5) korvautuu ehdolla $s_1 m_1 \mid h_1 m_1 2^{e_2} n'$, mikä on totta täsmälleen silloin, kun

$$s_1 = 2^{e_1} c_1 \mid h_1 2^{e_2} n'.$$

Koska $c_1 \mid n'$ ja luku n' on pariton, niin tämä on mahdollista jos ja vain jos

$$2^{e_1 - e_2} \mid h_1.$$

Vastaavasti ehto (3.6) on voimassa täsmälleen silloin, kun $2^{e_2} c_2 = s_2 \mid h_2 d = h_2 2^{e_2} n'$, eli yksinkertaistettuna $c_2 \mid h_2 n'$. Näin todella on, sillä $c_2 \mid n'$.

Nyt siis $a^{r_j d} \equiv b^d \equiv 1 \mod q_j, j = 1, 2$ täsmälleen silloin, kun $2^{e_1 - e_2} \mid h_1$. Jos siis $2^{e_1 - e_2} \mid h_1$ ja $n = q_1 q_2$, niin $\text{syt}(b^d - 1, n) = n$, jolloin b ei ole todistaja. Jos kuitenkin $2^{e_1 - e_2} \nmid h_1$, niin $b^d \not\equiv 1 \mod q_1$ ja $b^d \equiv 1 \mod q_2$, jolloin $1 < \text{syt}(b^d - 1, n) < n$ ja b on todistaja. Luvun h_1 kaikki parillisuustilanteet ovat yhtä yleisiä: puolessa tapauksista h_1 on parillinen, neljänneksessä tapauksista $2^2 = 4 \mid h_1$ ja niin edelleen. Nyt siis luvuista $b \in E_n$ enintään $\frac{1}{2^{e_1 - e_2}} \cdot \frac{\varphi(n)}{m_1 m_2} = \frac{\varphi(n)}{2^{e_1 - e_2} m_1 m_2} \leq \frac{\varphi(n)}{2m_1 m_2}$ ei ole todistajia, sillä $e_1 > e_2$ eli $e_1 \geq e_2 + 1$.

□

Lauseen 3.11 todistus. Olkoon $n > 4$ pariton yhdistetty luku. Jos $1 < b < n$ ja $\text{syt}(b, n) > 1$, niin $b^{n-1} \not\equiv 1 \mod n$; muutoin luvulla b olisi käänteisluku modulo n, mikä ei ole mahdollista näiden suurimman yhteisen tekijän ollessa ykköstä suurempi. Tällöin seurauksena $b \in W_n$. Todistaaksemme lauseen siis riittää osoittaa, että ainakin $\frac{3}{4}\varphi(n)$ luvuista $b \in E_n$ todistavat luvun n olevan yhdistetty, eli $b \in W_n$.

Jaamme todistuksen neljään osaan luvun n mahdollisten ominaisuuksien suhteen:

(1) $n = p^k$, missä $p \geq 3$ on alkuluku ja $k \geq 2$.

(2) On olemassa alkuluvut p_1, p_2 ja luvut $k_1, k_2 \geq 1$ siten, että $p_1^{k_1}, p_2^{k_2} \mid n$ ja $\varphi(p_1^{k_1}), \varphi(p_2^{k_2}) \nmid n-1$.
(3) On olemassa alkuluvut p_1, p_2 ja luvut $k_1, k_2 \geq 1$ siten, että $p_1^{k_1}, p_2^{k_2} \mid n$ ja $\varphi(p_1^{k_1}) \nmid n-1$, mutta $\varphi(p_2^{k_2}) \mid n-1$.
(4) Luvun $n = p_1^{k_1} \cdots p_r^{k_r}$ kaikille alkutekijöille $p_i, 1 \leq i \leq r$, pätee $\varphi(p_i^{k_i}) \mid n-1$.

Tapaus (1): Oletamme ensin, että $n = p^k, p \geq 3, k \geq 2$. Tällöin $\varphi(n) = p^{k-1}(p-1)$ ja $p \nmid n-1$, jolloin $p^{k-1} = \frac{\varphi(n)}{p-1} \leq \frac{\varphi(p^k)}{\mathrm{syt}(p^{k-1}(p-1), n-1)} = \frac{\varphi(p^k)}{\mathrm{syt}(\varphi(p^k), n-1)} =: m$. Lemman 3.13 nojalla enintään $\frac{\varphi(n)}{m}$ luvuista $b \in E_n$ toteuttaa yhtälön $b^{n-1} \equiv 1 \mod n$ ja siten $b \notin W_n$. Jos $n > 9$, niin koska $k \geq 2$, on oltava $p \geq 5$. Tällöin $4 < 5 \leq p^{k-1} \leq m$, jolloin väite seuraa. Jos $n = 9$, niin on vain kaksi kantalukua 1 ja 8, jotka eivät todista luvun n olevan yhdistetty.

Tapaus (2): Oletamme seuraavaksi, että luvulla n on ainakin kaksi eri alkulukutekijää p_1 ja p_2. Olkoon $q_i = p_i^{k_i}$, missä eksponentit k_i ovat mahdollisimman suuria siten, että $q_1, q_2 \mid n$. Jos $\varphi(q_1) \nmid n-1$, niin $2 \leq \frac{\varphi(q_1)}{\mathrm{syt}(\varphi(q_1), n-1)} =: m_1$, sillä $\mathrm{syt}(\varphi(q_1), n-1) < \varphi(q_1)$ ja selvästi $\mathrm{syt}(\varphi(q_1), n-1) | \varphi(q_1)$. Vastaavasti, jos $\varphi(q_2) \nmid n-1$, niin myös $m_2 := \frac{\varphi(q_2)}{\mathrm{syt}(\varphi(q_2), n-1)} \geq 2$. Jos nämä molemmat ehdot ovat voimassa, niin lemman 3.14 nojalla enintään $\frac{\varphi(n)}{m_1 m_2} \leq \frac{\varphi(n)}{4}$ luvuista $b \in E_n$ ei ole todistajien joukossa W_n.

Tapaus (3): Voimme nyt tehdä vastaavat merkinnät luvuille $q_i, m_i, i = 1, 2$, kuin tapauksessa (2) ja olettaa, että $\varphi(q_2) \mid n-1$ ja $\varphi(q_1) \nmid n-1$. Jos nyt $p_2 > 2$, niin luku

$$s_2 = \mathrm{syt}(\varphi(q_2), n-1) = \varphi(q_2) = p_2^{k_2-1}(p_2 - 1)$$

on parillinen ja lemman 3.14 jälkimmäisen tuloksen nojalla enintään $\frac{\varphi(n)}{2m_1 m_2} = \frac{\varphi(n)}{2m_1 \cdot 1} \leq \frac{\varphi(n)}{4}$ luvuista $b \in E_n$ ei ole todistajien joukossa W_n, sillä aiemman päätelmän nojalla oletus $\varphi(q_1) \nmid n-1$ johtaa siihen, että $m_1 \geq 2$. Jos taas $p_2 = 2$, niin parilliset luvut eivät ole suhteellisia alkulukuja luvulle n ja siten $\varphi(n) \leq \frac{n-1}{2}$, jolloin enintään

$$\frac{\varphi(n)}{m_1 m_2} = \frac{\varphi(n)}{m_1 \cdot \frac{2^{k_2-1}}{\mathrm{syt}(2^{k_2-1}, n-1)}} = \frac{\varphi(n)}{m_1 \cdot 2^{k_2-1}} \leq \frac{\varphi(n)}{m_1} \leq \frac{n-1}{2 \cdot m_1}$$

luvuista $1 \leq b < n$ toteuttaa yhtälön $b^{n-1} \equiv 1 \mod n$, sillä edelleen $m_1 \geq 2$.

Tapaus (4): Todistamme viimeisenä tapauksen, jossa luvulla $n = p_1^{k_1} \cdots p_r^{k_r}, r \geq 2$ on ainakin kaksi alkulukutekijää ja sille pätee $\varphi(p_i^{k_i}) \mid n-1$ kaikilla $1 \leq i \leq r$. Tällöin on oltava $k_i = 1$ kaikilla $1 \leq i \leq r$: Koska kahdelle mille tahansa luvun n alkulukutekijälle $p_i, p_j, 1 \leq i < j \leq r$ voidaan kirjoittaa $n = d p_i^{k_i} p_j^{k_j}$, missä luvut d, p_i ja p_j ovat toisilleen suhteellisia alkulukuja, niin $\varphi(p_i^{k_i}) = p_i^{k_i-1}(p_i - 1) \mid n-1$ ja vastaavasti $p_j^{k_j-1}(p_j - 1) \mid n-1$. Tällöin, lukujen $p_i^{k_i-1}$ ja $p_j^{k_j-1}$ ollessa suhteellisia alkulukuja, pätee

$$p_j^{k_j-1} p_j^{k_j-1} \mid n-1 = d p_i^{k_i} p_j^{k_j} - 1,$$

jolloin $dp_i^{k_i}p_j^{k_j} - 1 = cp_j^{k_j-1}p_j^{k_j-1}$ jollekin jakajalle $c \in \mathbb{N}$. Tämä on yhtäpitävää yhtälön $p_i^{k_i-1}p_j^{k_j-1}(dp_ip_j - c) = 1$, mikä on mahdollinen jos ja vain jos $p_j^{k_j-1}p_j^{k_j-1} = 1$, joka puolestaan toteutuu jos ja vain jos $k_i = k_j = 1$.

Helposti voimme huomata, että $r \geq 3$: muutoin $n = p_1p_2$, missä $p_1 < p_2$, ja

$$p_1 - 1, p_2 - 1 \mid n - 1 = p_1p_2 - 1 = p_1(p_2 - 1) + p_1 - 1,$$

mikä on mahdollista vain jos $p_2 - 1 \mid p_1 - 1$, mikä on ristiriidassa oletuksen $p_1 < p_2$ kanssa. Tällöin siis luvun n on oltava neliötön vähintään kolmen eri alkuluvun tulo ja jokaiselle alkulukujakajalle p on voimassa $p - 1 \mid n - 1$. Jos nyt valitsemme minkä tahansa luvun $a \in E_n$, niin luku a on suhteellinen alkuluku myös jokaiselle luvun n alkulukutekijälle p, jolloin $a^{p-1} \equiv 1 \mod p$. Koska $p-1 \mid n-1$, on voimassa myös yhtälö $a^{n-1} \equiv 1 \mod p$. Kaikki luvun n alkulukutekijät p ovat luonnollisesti suhteellisia alkulukuja toisilleen ja koska näille $p \mid a^{n-1} - 1$, niin mikä tahansa alkulukutekijöiden p tulo, mukaan lukien n, jakaa luvun $a^{n-1}-1$, jolloin $a^{n-1} \equiv 1 \mod n$.

Osoittamamme ominaisuuden nojalla luvun n on oltava Carmichaelin luku, jotka toteuttavat Fermat'n ehdon kaikilla kantaluvuilla $a \in E_n$. Olkoon nyt parittomat luvut p_1, p_2 ja p_3 kolme eri alkulukutekijää luvulle n siten, että $(p_i - 1) \mid (n - 1), i = 1, 2, 3$.

Oletamme seuraavaksi, että $e(p_1 - 1) = e(p_2 - 1) = e(p_3 - 1) =: e$. Koska $n - 1 = 2^f n'$ jollekin eksponentille $f \geq e$ ja parittomalle luvulle n', niin voimme valita luvun $d = 2^{e-1}n'$. Nyt $p_j - 1 \nmid d$, mutta $p_j - 1 \mid 2d$ kaikille $j = 1, 2, 3$. Voimme merkitä luvun $b \in E_n$ muodossa $b = a_j^{r_j} \mod p_j$, missä luvut a_j ovat primitiivisia juuria modulo p_j, kaikille $j = 1, 2, 3$. Nyt $b^d \equiv a_j^{r_jd} \equiv 1 \mod p_j$ täsmälleen silloin, kun $\varphi(p_j) = p_j - 1 \mid r_jd$ eli kun luku r_j on parillinen. Jos luku r_1 on parillinen ja luvut r_2, r_3 parittomia, niin $b^d \equiv 1 \mod p_1$ eli $p_1 \mid b^d - 1$, mutta $p_2, p_3 \nmid b^d - 1$, jolloin $1 < p_1 \leq \operatorname{syt}(b^d - 1, n) < n$. Vastaavaan tilanteeseen päädymme myös silloin, kun luvut r_1, r_2 ovat parillisia ja r_3 pariton. Elleivät siis luvut $r_i, i = 1, 2, 3$ ole kaikki parillisia tai kaikki parittomia, niin

$$1 < \operatorname{syt}(b^d - 1, n) < n,$$

jolloin $b \in W_n$. Lukujen r_j kaikkien pariteettien ollessa yhtä yleisiä ja pariteettimahdollisuuksien lukumäärän ollessa $2^3 = 8$ on nyt vain kaksi tapausta, joissa voi olla $b \notin W_n$, eli $\#W_n \geq (1 - \frac{2}{8})\varphi(n) = \frac{3}{4}\varphi(n)$.

Muut tapaukset menevät lähes vastaavasti: valitsemme alkulukujakajista $p_j, j = 1, 2, 3$ sen, jolle $e(p_j - 1)$ on pienin. Tekemättä oletuksia lukujen p_j suuruudesta voimme olettaa jakajan p_1 luvun $e(p_1 - 1) =: e_1$ olevan pienin. Tämän jälkeen valitsemme luvun $d = 2^{e_1-1}n'$, jolloin $p_1 - 1 \nmid d$, mutta $p_1 - 1 \mid 2d$. Nyt ainakin yhdelle $p_k, k \neq 1$ pätee $p_k - 1 \nmid d$ ja $p_k - 1 \nmid 2d$, mutta $p_k - 1 \mid 2^{e(p_k-1)-e_1+1}d = 2^{e(p_k-1)}n'$. Voimme olettaa, että $k = 3$. Lisäksi nyt luvulle p_2 pätee $p_2 - 1 \nmid d$, mutta $p_2 - 1 \mid 2^{e(p_2-1)-e_1+1}d$, missä $e_1 \leq e(p_2 - 1) < e(p_3 - 1)$. Olkoon jatkossa merkintöjen lyhentämisen vuoksi $e_2 := e(p_2 - 1)$ ja $e_3 := e(p_3 - 1)$.

Voimme jälleen valita primitiiviset juuret a_j modulo $p_j, j = 1, 2, 3$ siten, että voimme merkitä luvun $b \in E_n$ muodossa $b = a_j^{r_j} \mod p_j$ joillekin eksponenteille $r_j, j = 1, 2, 3$. Nyt $1 < \text{syt}(b^d - 1, n) < n$ vain, jos ei päde $\text{syt}(b^d - 1, n) = n$ eikä $\text{syt}(b^d - 1, n) = 1$. Yhtälö $\text{syt}(b^d - 1, n) = n$ voi olla voimassa, jos $n = p_1p_2p_3$ ja

$$b^d \equiv a^{r_j d} \equiv 1 \mod p_j \text{ kaikille } j = 1, 2, 3,$$

mikä pätee täsmälleen silloin, kun $p_j - 1 \mid r_j d$. Näin puolestaan on täsmälleen silloin, kun luvuilla $r_j, j = 1, 2, 3$ on sopiva pariteetti, eli $2 \mid r_1$, $2^{e_2-e_1+1} \mid r_2$ ja $2^{e_3-e_1+1} \mid r_3$. Olkoon jatkossa merkintöjen lyhentämiseksi luvut $t := e_2 - e_1 + 1$ ja $s := e_3 - e_1 + 1$. Koska kaikki pariteettitapaukset ovat yhtä yleisiä, niin näin pätee vain yhdessä tapauksessa $2 \cdot 2^t \cdot 2^s$ tapauksesta. Yhtälö $\text{syt}(b^d - 1, n) = 1$ on puolestaan mahdollinen ainoastaan silloin, kun

$$b^d \equiv a^{r_j d} \not\equiv 1 \mod p_j \text{ kaikille } j = 1, 2, 3,$$

eli kun $2 \nmid r_1$, $2^t \nmid r_2$ ja $2^s \nmid r_3$. Näin on jälleen vain yhdessä tapauksessa $2 \cdot 2^t \cdot 2^s$ tapauksesta. Kaiken kaikkiaan $b \notin W_n$ enintään $\frac{1+1}{2 \cdot 2^t \cdot 2^s} \leq \frac{2}{16} = \frac{1}{8}$ tapauksista, sillä $s > 1$ ja $t \geq 1$. Tällöin siis $W_n \geq (1 - \frac{1}{8})\varphi(n) = \frac{7}{8}\varphi(n) \geq \frac{3}{4}\varphi(n)$.

□

LAUSE 3.15. *Seuraavat ehdot parittomalle luvulle $n > 4$ ja luvulle $1 \leq a \leq n - 1$ ovat ekvivalentteja:*

(1) $a \in W_n$
(2) *Luku n ei läpäise Millerin ja Rabinin testin ehtoa* (3.1) *eikä* (3.2) *kantaluvulla a.*

TODISTUS. Olkoon nyt $s = \max\{l : 2^l \mid n-1\}$ ja d pariton siten, että $n-1 = 2^s d$. Täsmennämme aluksi ehtoa $a \in W_n$: Todistajien määritelmän nojalla luvun a ollessa todistaja on voimassa $a^{n-1} \not\equiv 1 \mod n$ tai jollekin $i = 0, 1, ..., s$ pätee

$$1 < \text{syt}(a^{2^i d} - 1, n) < n.$$

Tällöin luvulle a joko pätee $a^{n-1} \not\equiv 1 \mod n$ tai jollekin $i = 0, 1, ..., s - 1$ pätee $1 < \text{syt}(a^{2^i d} - 1, n) < n$: jos molemmat ehdot

$$1 < \text{syt}(a^{2^s d} - 1, n) = \text{syt}(a^{n-1} - 1, n) < n$$

ja

$$a^{n-1} \equiv 1 \mod n$$

olisivat voimassa, niin tällöin $n \mid a^{n-1} - 1$, mutta $\text{syt}(a^{n-1} - 1, n) < n$.

Määrittelemme suurimpien yhteisten tekijöiden jonon $(S_i)_{i=0}^s$ siten, että

$$S_i = \text{syt}(a^{2^i d} - 1, n).$$

Jos $a \notin W_n$, niin koska $a^{n-1} \equiv 1 \mod n$, on oltava $S_s = n$. Jos jollekin $i = 0, 1, ..., s-1$ on $S_i = n$, niin $a^{2^i d} \equiv 1 \mod n$. Neliöimällä tämän yhtälön molemmat puolet saamme yhtälön

$$a^{2^{i+1} d} \equiv 1 \mod n,$$

jolloin $S_{i+1} = n$. Voimme siis huomata, että tiedosta $S_i = n$ seuraa, että $S_{i+1} = n$. Nyt siis luvulle $a \notin W_n$ jonon (S_i) on oltava joko muotoa

(1) $S_i = n$ kaikille $0 \leq i \leq s$ tai
(2) $S_i = 1$ kaikille $0 \leq i \leq i_0$ ja $S_i = n$ kaikille $i_0 < i \leq s$ jollekin indeksille $i_0 \in \{0, 1, ..., s-1\}$;

jos olisi $1 < S_i = \text{syt}(a^{2^i d} - 1, n) < n$ jollekin $i = 0, 1, ..., s-1$, niin selvästi a olisi todistaja. Ensimmäinen tapaus on ekvivalentti sille, että $a^d \equiv 1 \mod n$, jolloin Millerin ja Rabinin testin ehto (3.1) toteutuu. Jälkimmäisessä tapauksessa $\text{syt}(a^{2^{i_0} d} - 1, n) = 1$, jolloin $a^{2^{i_0} d} \not\equiv 1 \mod n$. Kuitenkin $\text{syt}(a^{2^{i_0+1} d} - 1, n) = n$, jolloin

$$a^{2^{i_0+1} d} = (a^{2^{i_0} d})^2 \equiv 1 \mod n.$$

Tällöin siis

$$(a^{2^{i_0} d})^2 - 1 = (a^{2^{i_0} d} - 1)(a^{2^{i_0} d} + 1) \equiv 0 \mod n,$$

missä luvut n ja $(a^{2^{i_0} d} - 1)$ ovat suhteellisia alkulukuja, joten

$$n \mid (a^{2^{i_0} d} + 1) \text{ ja } a^{2^{i_0} d} \equiv -1 \mod n.$$

Tällöin siis Millerin ja Rabinin testin ehto (3.2) toteutuu.

Nyt siis olemme todistaneet, että jos luku a ei ole todistaja luvulle n, niin Millerin ja Rabinin testin ehdoista ainakin toinen on voimassa. Tämä on yhtäpitävää sen kanssa, että jos kumpikaan Millerin ja Rabinin testin ehdoista ei ole voimassa, niin a on todistaja luvulle n.

Todistamme seuraavaksi väitteen toiseen suuntaan, eli jos luku a on todistaja luvulle n, niin kumpikaan Millerin ja Rabinin testin ehdoista ei ole voimassa kantaluvulle a. Väitteen todistamiseksi riittää osoittaa, että jos ainakin yksi Millerin ja Rabinin testin ehdoista (3.1) tai (3.2) on voimassa kantaluvulle a, niin luku a ei voi olla todistaja luvulle n.

Millerin ja Rabinin ehdoista (3.1) ja (3.2) voi olla vain yksi kerrallaan voimassa kantaluvulle a: Jos $a^d \equiv 1 \mod n$, niin neliömällä tämän yhtälön molemmat puolet $s-1$ kertaa on selvää, että ei voi olla $a^{2^i d} \equiv -1 \mod n$ millekään $i = 0, 1, ..., s-1$, ellei $n = 2$. Oletimme kuitenkin, että $n > 4$.

Oletamme ensin, että ehto (3.1) on voimassa, eli $a^d \equiv 1 \mod n$. Tällöin neliöimällä s kertaa saamme yhtälön $a^{n-1} \equiv 1 \mod n$, jolloin tältä osin a ei voi olla todistaja luvulle n. Lisäksi neliöimällä voimme huomata, että kaikille $i = 0, 1, ..., s$ on voimassa yhtälö $a^{2^i d} \equiv 1 \mod n$, jolloin $n \mid a^{2^i d} - 1$ ja siten $\text{syt}(a^{2^i d} - 1, n) = n$, eli luku a ei täytä todistajan ehtoja lainkaan.

Jos ehto (3.2) on voimassa, niin jollekin $i_0 \in \{0, 1, ..., s-1\}$ on voimassa yhtälö $a^{2^{i_0} d} \equiv -1 \mod n$. Neliöimällä kyseistä yhtälöä $s - i_0$ kertaa saamme yhtälön

$$a^{2^s d} = a^{n-1} \equiv 1 \mod n.$$

Nyt kaikille $i > i_0$ siis $a^{2^i d} \equiv 1 \mod n$, jolloin $n \mid a^{2^i d} - 1$ ja siten $\text{syt}(a^{2^i d} - 1, n) = n$. Jos jollekin $i_1 \in \{0, 1, ..., i_0\}$ on voimassa $1 < \text{syt}(a^{2^{i_1} d} - 1, n) < n$, niin voimme merkitä $m := \text{syt}(a^{2^i d} - 1, n)$ ja pätee $a^{2^{i_1} d} \equiv 1 \mod m$. Neliöimällä tämän yhtälön

$i_0 - i_1$ kertaa saamme

$$a^{2^{i_0}d} \equiv 1 \mod m.$$

Oletuksemme nojalla pätee $a^{2^{i_0}d} \equiv -1 \mod n$ ja koska $m \mid n$, niin

$$a^{2^{i_0}d} \equiv -1 \mod m.$$

Tällöin siis on oltava $-1 \equiv 1 \mod m$, mikä on mahdollista vain jos $m = 2$. Oletimme kuitenkin, että n on pariton, jolloin m ei voi olla sen jakaja ja päädymme ristiriitaan. Tässäkään tapauksessa a ei siis voi olla todistaja luvulle n, joten olemme todistaneet viimeisenkin osan väitteestä.

□

Nyt lauseen 3.9 todistamiseksi riittää yhdistää aiemmat päätelmät:

Lauseen 3.9 todistus. Lauseen 3.11 nojalla yhdistetylle luvulle n todistajien joukon W_n suuruus on ainakin $\frac{3}{4}(n-1)$ alkiota luvista $1, 2, ..., n-1$. Jos luku a on todistajien joukossa luvulle n, niin koska oletimme luvun n olevan pariton, voimme todeta lauseen 3.15 nojalla, ettei luku a toteuta kumpaakaan ehdoista (3.1) tai (3.2) ollessaan kantalukuna. Siispä lukujen, joille Millerin ja Rabinin testi ehdot eivät ole voimassa, osuus on ainakin $\frac{3}{4}$, jolloin enintään $\frac{1}{4}$ kantaluvuista $a \in \{1, 2, ..., n-1\}$ toteuttaa jommankumman ehdoista (3.1) tai (3.2). □

Huomautus 3.16. Varsinaisessa Millerin ja Rabinin algoritmissa ei ole oleellista tutkia Fermat'n ehdon paikkansapitävyyttä laskemalla: Fermat'n ehdon luvun $a^{n-1} \mod n$ neliöjuuresta $a^{2^{s-1}d}$ on pääteltävissä myös Fermat'n ehdon sekä ehdon (3.2) pitävyys tutkimalla onko se -1 vai jotain muuta.

Huomautus 3.17. On syytä huomata, että jos jossain vaiheessa neliöintien modulo n aikana saamme tulokseksi 1, niin luvun -1 ilmestyminen seuraavien neliöintien aikana on mahdotonta, ja siten n on yhdistetty luku. Voimme käyttää tätä tietoa vähentääksemme varsinaisen testin laskutoimitusten lukumäärää.

Nyt olemme saavuttaneet viimeinkin rajan, jonka avulla voimme arvioida testin luotettavuutta.

3.1.3. Millerin ja Rabinin algoritmi. Millerin ja Rabinin alkulukutesti on luonteeltaan probabilistinen. Lauseen 3.9 raja-arvosta 1/4 saamme arvion testin luotettavuudesta: jos testaamme yhdistettyä lukua n satunnaisesti eri kantaluvuilla $a = 2, 3, ..., n-2$, niin jokaisella kierroksella todennäköisyys osua todistajaan - eli lukuun, jolla kumpikaan Millerin ja Rabinin ehdoista (3.1) ja (3.2) ei toteudu - on ainakin 3/4. Jos luku n on yhdistetty, niin k testin jälkeen se on *todennäköisesti alkuluku* enintään todennäköisyydellä $\frac{1}{4^k}$, jota kutsumme testin virherajaksi.

Kuvailemme algoritmin sanallisesti seuraavasti: olkoon $n > 4$ testattava (pariton) luku, jota testaamme k kertaa.

(1) Laskemme luvut $s = \max\{j \in \mathbb{N} : 2^j \mid n-1\}$ ja $d = n/2^s$.
(2) Suoritamme k kertaa:
    (i) Valitsemme satunnaisen luvun $a \in \{2, 3, ..., n-2\}$.
    (ii) Laskemme $A := a^d \mod n$.
    (iii) Jos $A = 1$ tai $A = -1$, niin palaamme vaiheen (2) alkuun.

(iv) Suoritamme $s-1$ kertaa:
$A := A^2 \mod n$.
Jos $A = 1$ palautamme *yhdistetty luku*.
Jos $A = -1$, niin palaamme vaiheen (2) alkuun.
(v) Palautamme *yhdistetty luku*.
(3) Palautamme *todennäköisesti alkuluku*.

Algoritmi palauttaa siis ”todennäköisesti alkuluku” ja ohjelmakoodiversiossa ”tosi” (true), jos testattava luku n läpäisee kaikki k testiä, jolloin luku n on todettavissa todennäköisesti alkuluvuksi. Jos yhdelläkin kierroksella testattava luku ei toteuta kumpaakaan ehdoista (3.1) tai (3.2), niin palautusarvo on ”yhdistetty luku”, ohjelmakoodissa ”epätosi” (false).

Saamme helposti muodostettua esimerkiksi seuraavanlaisen ohjelmakoodin:

```
// Millerin ja Rabinin algoritmi
static bool onTodennakoisestiAlkuluku_MR(BigInteger n, int
   kierroksia)
{
    if (n == 2) return true;
    if (n.IsEven) return false;

    BigInteger d = n-1;
    int s = 0;
    while (d.IsEven)
    {
            d = d / 2;
            s++;
    }
    Random satunnainen = new Random();
    int pituus = n.ToByteArray().Length;
    byte[] tavut = new byte[pituus];

    BigInteger x;
    BigInteger a;
    for (int i = 0; i < kierroksia; i++)
    {
            // Arvomme kantaluvun väliltä [2,n-2]
            satunnainen.NextBytes(tavut);
            a = BigInteger.Abs(new BigInteger(tavut)) % (n-3)
               + 2;
            // Laskemme a^d mod n ja tarkastamme
               jakojäännöksen
            x = BigInteger.ModPow(a, d, n);
            if (x == 1 || x == n - 1) continue;
            // Jatkamme tarvittaessa eteenpäin
            int j;
```

```
            for (j = 0; j < s-1; j++)
            {
                x = BigInteger.ModPow(x, 2, n);
                if (x == 1) return false;
                if (x == n - 1) break;
            }
            if (j >= s - 1) return false;
        }

        return true;
    }
```

Vaikka Millerin ja Rabinin testi näyttää nopeasti katsottuna hyvinkin paljon Fermat'n testiä monimutkaisemmalta, niin osoittautuu, että molemmat toimivat suurin piirtein samassa ajassa. Ensimmäisenä algoritmia suorittaessamme hylkäämme kaikki kakkosta suuremmat parilliset luvut. Seuraavana laskemme, kuinka monesti luku $n-1$ on jaettavissa kahdella; pahimmassa tapauksessa testattava luku n on muotoa 2^l+1, jolloin $d=1$ ja jakolaskuja ja yhteenlaskuja tulee olemaan $l=\lfloor \log_2 n \rfloor$ kappaletta. Tähän mennessä siis olemme tehneet laskutoimituksia $\mathcal{O}(\log n \log 2) + \mathcal{O}(\log_2 n(\log n \log 2 + \log n)) = \mathcal{O}(\log^2 n)$.

Seuraavaksi pystymme vakioajassa $\mathcal{O}(1)$ luomaan tavupuskurin sekä tarvittavat muuttujat sekä suorittamaan useita muita laskennallisesti kevyempiä vaiheita. Silmukassa arvomme k kertaa kantaluvun a, jonka myös skaalaamme välille $[2, n-2]$ kolmella laskutoimituksella, mikä on kertaluokaltaan suurin piirtein $\mathcal{O}(\log n \log n + \log n) = \mathcal{O}(\log^2 n)$. Lisäksi teemme potenssiinkorotuksia modulo n aiemmin käsittelemällämme toistetulla neliöinnillä, jonka aikavaativuus tässä yhteydessä on $\mathcal{O}(\log^3 n)$. Pahimmassa tapauksessa tämän jälkeinen ehtolauseen arviointi ei lopeta silmukan kierrosta, vaan joudumme siirtymään sisempään silmukkaan tutkimaan, päteekö $x^{2^r} \equiv -1 \mod n$ millään luvuista $r = 1, ..., s-1$. Sisempi silmukka pyörähtää enintään $l = \lfloor \log_2 n \rfloor$ kertaa ja siihen sisältyy jokaisella kierroksella $\mathcal{O}(\log^2 n)$-kertolasku ja $\mathcal{O}(\log^2 n)$-jakojäännöslasku. Sisempi silmukka on siis kertaluokaltaan

$$\lfloor \log_2 n \rfloor \cdot (\mathcal{O}(\log^2 n) + \mathcal{O}(\log^2 n)) = \mathcal{O}(\log^3 n).$$

Pääsilmukkaa käymme läpi ennalta päätetyt k kertaa, jolloin algoritmin kokonaissuoritusajaksi saamme

$$\mathcal{O}(\log^2 n) + k \cdot (\mathcal{O}(\log^2 n) + \mathcal{O}(\log^3 n) + \mathcal{O}(\log^3 n)) = \mathcal{O}(k \log^3 n).$$

Helposti voimme huomata, että tehostamalla kerto- ja jakolaskua myös tämän testin suoritusaika putoaa luokkaan $\mathcal{O}^{\sim}(k \log^2 n)$.

Rabinin 1/4-ehdon nojalla testistä voi helposti muokata deterministisen: riittää, että testaamme kaikki kantaluvut a joukosta $\left\{2, 3, ..., \lceil \frac{n}{4} \rceil\right\}$. Tällöin käymme läpi kierroksia enintään $n/4$ kappaletta, jolloin algoritmin deterministisen variantin suoritusajaksi saamme kertaluokan $\mathcal{O}(\frac{n}{4} \log^3 n) = \mathcal{O}(n \log^3 n)$. Deterministisen variantin heikkoutena on kuitenkin suuri n-kerroin: käytännön kannalta on hyvin todennäköistä, että lukua n ei todeta virheellisesti todennäköiseksi alkuluvuksi, jos testejä tehdään

esimerkiksi $k = 1000$ kappaletta, mutta testattavan luvun neljännes $n/4$ on huomattavasti suurempi luvun n ollessa esimerkiksi yli satanumeroisten lukujen kokoluokassa. Deterministinen versio Millerin ja Rabinin testistä häviää itse asiassa selkeästi jopa tehottomalle jakolaskumenetelmälle, jonka kertaluokaksi totesimme $\mathcal{O}(\sqrt{n}\log^2 n)$.

3.2. Solovayn ja Strassenin testi

3.2.1. Jacobin symbolit ja Solovayn ja Strassenin algoritmi. Solovayn ja Strassenin testi on eräs käytännöllisistä probabilistisista alkulukutesteistä: se ei vaadi laskennallisesti pitkää suoritusaikaa. Algoritmin oleellisin osuus perustuu *Legendren symbolin* arvoon, joten muistelemme aluksi sen määritelmää. Ennen määritelmää on tulee kuitenkin määritellä käsitteet *neliöjäännös* ja *epäjäännös*: sanomme, että luku a on neliöjäännös modulo p, jos kongruenssiyhtälöllä $x^2 \equiv a \mod p$ on ratkaisu. Muutoin luku a on epäjäännös, paitsi jos $a \in [0]_p$, jolloin luku a ei ole neliöjäännös eikä epäjäännös. Alkuluvulle p niin epä- kuin neliöjäännöksiä modulo p on $\frac{p-1}{2}$ kappaletta [**11**, s. 51].

Määritelmä 3.18. Olkoon $p > 2$ alkuluku. Tällöin määrittelemme Legendren symbolin

$$\left(\frac{a}{p}\right) = \begin{cases} 1, \text{ jos } a \text{ on neliöjäännös modulo } p \\ -1, \text{ jos } a \text{ on epäjäännös modulo } p \\ 0, \text{ jos } a \equiv 0 \mod p. \end{cases}$$

Legendren symbolilla on lukuisia laskusääntöjä, joiden avulla sen arvon selvittäminen käy kohtalaisessa ajassa. Jatkon kannalta yleistämme käsitteen parittomille yhdistetyille luvuille: Jacobin symboli $\left(\frac{a}{n}\right)$ on täsmälleen sama kuin Legendren symboli, kun $n > 2$ on alkuluku, mutta parittomille yhdistetylle luvulle $n = p_1^{\alpha_1} \cdot \ldots \cdot p_k^{\alpha_k}$ määrittelemme

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{\alpha_1} \cdot \ldots \cdot \left(\frac{a}{p_k}\right)^{\alpha_k}.$$

Jacobin symboleille pätevät pikälti samanlaiset ominaisuudet kuin Legendren symboleille, mutta se ei enää määritelmänsä vuoksi ilmoita, onko luku neliöjäännös. Hyödynnämme näiden laskusääntöjä myöhemmin tutkiessamme algoritmin rakennetta.

Lause 3.19. *Jacobin symbolille* $\left(\frac{a}{n}\right)$ *pätee seuraavaa:*

(1) *Jos* $a \equiv b \mod n$, *niin* $\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$.

(2) $\left(\frac{a}{n}\right) = \begin{cases} 0, \textit{ jos } \mathrm{syt}(a,n) > 1 \\ \pm 1, \textit{ jos } \mathrm{syt}(a,n) = 1 \end{cases}$

(3) $\left(\frac{a}{n}\right) \cdot \left(\frac{b}{n}\right) = \left(\frac{ab}{n}\right)$

(4) $\left(\frac{a}{m}\right) \cdot \left(\frac{a}{n}\right) = \left(\frac{a}{mn}\right)$

(5) *Jos* $\mathrm{syt}(m,n) = 1$, *niin*

$$\left(\frac{m}{n}\right) = \begin{cases} -\left(\frac{n}{m}\right), \textit{ jos } m \equiv n \equiv 3 \mod 4 \\ \left(\frac{n}{m}\right), \textit{ muuten} \end{cases}$$

(6) $\left(\frac{-1}{n}\right) = \begin{cases} 1, \text{ jos } n \equiv 1 \mod 4 \\ -1, \text{ jos } n \equiv 3 \mod 4 \end{cases}$

(7) $\left(\frac{2}{n}\right) = \begin{cases} 1, \text{ jos } n \equiv 1 \text{ tai } n \equiv 7 \mod 8 \\ -1, \text{ jos } n \equiv 3 \text{ tai } n \equiv 5 \mod 8. \end{cases}$

(8) *Jos* $\left(\frac{a}{n}\right) = -1$, *niin* a *on epäjäännös modulo* n.

TODISTUS. Sivuutamme todistuksen liian pitkänä tavoitteemme kannalta, mutta todistus on löydettävissä lähdemateriaalista [**11**, Luku 4, ss. 50-78]. □

Seuraava lauseen tulos muodostaa pohjan Solovayn ja Strassenin testille:

LAUSE 3.20. *Olkoon* $p > 2$ *alkuluku. Tällöin Legendren symbolille* $\left(\frac{a}{p}\right)$ *pätee yhtälö*

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \mod p. \tag{3.7}$$

TODISTUS. Jos a on neliöjäännös modulo p eli $a \equiv x^2 \mod p$ jollekin $x \in \mathbb{Z}_{\mathrm{p}}$, niin $\left(\frac{a}{p}\right) = 1$ ja

$$a^{\frac{p-1}{2}} \equiv (x^2)^{\frac{p-1}{2}} = x^{p-1} \equiv 1 \mod p$$

Fermat'n pienen lauseen nojalla. Jos $a \equiv 0 \mod p$, niin selvästi yhtälö on voimassa. Rittää siis osoittaa yhtälön pitävyys tapauksessa, jossa a on epäjäännös modulo p.

Olkoon luku $b \in \{1, 2, ..., p-1\}$. Lineaarisella kongruenssiyhtälöllä $bx \equiv a \mod p$ on täsmälleen yksi ratkaisu $b' \neq b$, sillä oletimme luvun a olevan epäjäännös ja koska p on alkuluku, pätee $\mathrm{syt}(b, p) = 1$ [**11**, s. 23]. Koska b on valittavissa mielivaltaisesti, voimme järjestää luvut $\{1, 2, ..., p-1\}$ mielivaltaisesti $\frac{p-1}{2}$ kappaleeksi pareja (b, b') siten, että $bb' \equiv a$. Näiden kaikkien tulon on oltava

$$(p-1)! = (p-1)(p-2) \cdot ... \cdot 2 \cdot 1 \equiv \underbrace{a \cdot a \cdot ... \cdot a}_{\frac{p-1}{2} \text{ kappaletta}} = a^{\frac{p-1}{2}} \mod p.$$

Aiemmin käsittelemämme Wilsonin lauseen nojalla $(p-1)! \equiv -1 \mod p$, jolloin $a^{\frac{p-1}{2}} \equiv -1 \equiv \left(\frac{a}{p}\right) \mod p$, sillä a oli tässä epäjäännös. □

Kääntäen saamamme tuloksen voimme huomata, että jos yhtälö (3.7) on totta joillekin luvuille a ja p, niin p on alkuluku. Sen sijaan saman yhtälön toteutuminen Jacobin symbolille, jossa käytämme alkuluvun p sijaan mielivaltaista paritonta lukua n, ei tarkoita luvun n olevan alkuluku. Yhtälön (3.7) toteutumisen tarkistaminen osoittautuu kuitenkin varsin hyväksi keinoksi alkulukutestauksessa, sillä parittomille yhdistetyille luvuille $n > 3$ saamme muodostettua todennäköisyyden, jolla luku n valehtelee olevansa alkuluku.

Jotta saamme arvioitua todennäköisyyttä, tarvitsemme joukon

$$E(n) = \left\{ a \in \mathbb{Z}_n^\times \ : \ \left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}} \mod n \right\}$$

sekä seuraavan lemman:

LEMMA 3.21. *Olkoon luku $n \geq 3$ pariton. Tällöin n on alkuluku jos ja vain jos $E(n) = \mathbb{Z}_n^\times$.*

TODISTUS. Joukon $E(n)$ määritelmän nojalla $E(n) \subset \mathbb{Z}_n^\times$, joten riittää osoittaa, että inkluusio pätee toiseen suuntaan. Jos $n \geq 3$ on alkuluku, niin jokainen $a \in \mathbb{Z}_n^\times$ lauseen 3.20 nojalla on myös joukossa $E(n)$. Tällöin siis $\mathbb{Z}_n^\times \subset E(n)$, joten väite on tässä tapauksessa totta.

Jos $n \geq 3$ on yhdistetty luku, niin oletamme vastoin väitettä, että $\mathbb{Z}_n^\times \subset E(n)$. Tällöin jokaiselle $a \in \mathbb{Z}_n^\times$ on oltava

$$a^{n-1} \equiv (a^{\frac{n-1}{2}})^2 \equiv (\pm 1)^2 \equiv 1 \mod n.$$

Nyt luku n on Carmichaelin luku ja siten lauseen 3.6 nojalla neliötön, jolloin $n = dp$, missä p on alkuluku ja $\text{syt}(d, p) = 1$. Koska alkuluvulle p luvuista $a = 1, 2, ..., p-1$ puolet on epäjäännöksiä, niin voimme valita luvun b, joka on epäjäännös modulo p. Edelleen kiinalaisen jäännöslauseen nojalla voimme valita luvun $a \in \mathbb{Z}_n^\times$ siten, että $a \equiv b \mod p$ ja $a \equiv 1 \mod d$. Legendren symbolin määritelmän nojalla $\left(\frac{b}{p}\right) = -1$, jolloin Jacobin symbolin määritelmän ja luvun a valinnan nojalla

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p}\right)\left(\frac{a}{d}\right) = \left(\frac{b}{p}\right) \cdot 1 = -1.$$

Koska $a \in E(n)$, on nyt oltava $a^{\frac{n-1}{2}} \equiv -1 \mod n$. Koska $r \mid n$, niin pätee myös $a^{\frac{n-1}{2}} \equiv -1 \mod r$, mutta tämä on ristiriita, sillä $a \equiv 1 \mod r$. □

LAUSE 3.22. *Jos n on pariton yhdistetty luku, niin yhtälö* (3.7) *on voimassa korkeintaan puolella luvuista $a \in \{1, 2, ..., n-1\}$.*

TODISTUS. Lemman 3.21 ja joukon $E(n)$ määritelmästä seuraa, että $E(n)$ on joukon $\mathbb{Z}_n^\times$ aliryhmä: Jos $a, b \in E(n)$, niin $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right) \cdot \left(\frac{b}{n}\right) \equiv a^{\frac{n-1}{2}} b^{\frac{n-1}{2}} = (ab)^{\frac{n-1}{2}} \mod n$. Koska $b \in \mathbb{Z}_n^\times$, niin luvulla b on käänteisalkio b^{-1} ja näille pätee

$$\left(\frac{b}{n}\right)\left(\frac{b^{-1}}{n}\right) = \left(\frac{bb^{-1}}{1}\right) = \left(\frac{1}{n}\right) = 1 = 1^{\frac{n-1}{2}} \equiv (bb^{-1})^{\frac{n-1}{2}} = b^{\frac{n-1}{2}}(b^{-1})^{\frac{n-1}{2}} \mod n.$$

Koska $\left(\frac{b}{n}\right) \equiv b^{\frac{n-1}{2}} \mod n$, niin ryhmän $\mathbb{Z}_n^\times$ supistussäännön nojalla $\left(\frac{b^{-1}}{n}\right) \equiv (b^{-1})^{\frac{n-1}{2}} \mod n$, jolloin myös $b^{-1} \in E(n)$.

Nyt Lagrangen lauseen nojalla $\#E(n) = \text{ord}\, E(n) \mid \text{ord}\, \mathbb{Z}_n^\times = \#\mathbb{Z}_n^\times$, joten $\#\mathbb{Z}_n^\times = d \cdot \#E(n)$, missä $d \geq 2$, sillä $E(n)$ on aito osajoukko joukolle $\mathbb{Z}_n^\times$. Helposti voimme tällöin päätellä, että

$$\#E(n) \leq \frac{\#\mathbb{Z}_n^\times}{2} = \frac{\varphi(n)}{2} \leq \frac{n-1}{2}.$$

□

Koska olemme saaneet nyt selville, miten usein pahimmassa tapauksessa ehto (3.7) pitää parittomille yhdistetyille $n \geq 3$, niin voimme muodostaa Millerin ja Rabinin testin kaltaisen probabilistisen testin. Testiä kutsutaan yleisesti Solovayn ja Strassenin

testiksi.

Solovayn ja Strassenin testissä valitsemme luvun a satunnaisesti joukosta $\{2, 3, ..., n-1\}$ ja testaamme parittomalle luvulle n yhtälön

$$\left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}} \mod n \tag{3.8}$$

pitävyyttä. Testattuamme k kappaletta kantalukuja a ja ehdon (3.8) pitäessä näillä kaikilla kerroilla, voimme todeta luvun n olevan todennäköisesti alkuluku. Jos n on oikeasti yhdistetty luku, niin virhe on tehty todennäköisyydellä $\left(\frac{1}{2}\right)^k$. Kuten Millerin ja Rabinin testissä, yksikin todistaja a riittää osoittamaan luvun n varmasti yhdistetyksi.

Tarvitsemme vielä tavan laskea Legendren symbolin $\left(\frac{a}{n}\right)$ arvo tehokkaasti. Lauseen 3.19 ehdot mahdollistavat yllättävän toimivan laskemisalgoritmin Legendren symbolille, sillä sen arvo on sama kuin vastaavan Jacobin symbolin. Nimitämme jatkossa Jacobin symbolin viivan yläpuolista osaa osoittajaksi ja alapuolta nimittäjäksi, kuten murtoluvuille. Laskentatapa toimii seuraavasti:

(1) Jos osoittaja a ja nimittäjä n eivät ole suhteellisia alkulukuja, palautamme 0 (sääntö 2).
(2) Vähennämme osoittajaa a modulo n (sääntö 1).
(3) Tarkistamme, kuinka monesti osoittaja a on jaollinen kahdella ja poistamme siitä kaikki kertoimet 2. Uudelle symbolille lisäämme kertoimen -1, mikäli a jakautuu kahdella parittoman määrän kertoja ja $n \equiv 3$ tai $5 \mod 8$. Muutoin kerroin on 1 (säännöt 3 ja 7).
(4) Jos osoittaja $a = 1$, niin palautamme vaiheessa 2 saamamme kertoimen ja ykkösen tulon - lyhyesti siis pelkän kertoimen (koska 1 on triviaalisti neliöjäännös).
(5) Vaihdamme osoittajan a ja nimittäjän n paikkoja. Jos $a \equiv n \equiv 3 \mod 4$, niin lisäämme uudelle symbolille kertoimen -1, muutoin kerroin on 1 (sääntö 5). Laskemme uuden symbolin arvon siirtymällä vaiheeseen 2. Palautamme uuden symbolin sekä vaiheissa 3 ja 5 saamamme kertoimien tulon.

Nyt voimme helposti johtaa ohjelmakoodin niin varsinaisesta Solovayn ja Strassenin testistä kuin Jacobin symbolin laskemisesta. Mukaan on liitettävä rekursiivinen toteutus Eukleideen algoritmille, jotta suurin yhteinen tekijä olisi helposti tarkistettavissa.

```
// Probabilistinen Solovayn ja Strassenin alkulukutesti
static bool onTodennakoisestiAlkuluku_SolovayStrassen
(BigInteger n, int k)
{
  Random r = new Random();
  byte[] tavut = new byte[n.ToByteArray().Length];
  BigInteger a;
  for (int i = 0; i < k; i++)
  {
    r.NextBytes(tavut);
```

```
    a = BigInteger.Abs(new BigInteger(tavut)) % (n - 1) + 1;
    BigInteger jacobinSymboli = JacobinSymboli(a, n);
    if (jacobinSymboli < 0) jacobinSymboli += n;
    if (jacobinSymboli != BigInteger.ModPow(a, (n - 1) / 2,
       n)) return false;
  }
  return true;
}

// Laskee Jacobin symbolin (a|n) arvon
static int JacobinSymboli(BigInteger a, BigInteger n)
{
    if (syt(a, n) > 1) return 0;
    else return JacobinSymboliSyt1(a, n);
}

// Laskee Jacobin symbolin (a|n) arvon rekursiivisesti, jos
// a ja n ovat suhteellisia alkulukuja
static int JacobinSymboliSyt1(BigInteger a, BigInteger n)
{
  int kerroin = 1;
  a = a % n;
  if (a == 0) return 0;
  BigInteger s;
  for (s = 0; a % 2 == 0; a = a / 2, s++) ;

  if ((s % 2 == 1) && (n % 8 == 3 || n % 8 == 5))
    kerroin = -kerroin;

  if (a == 1) return kerroin;

  if (a % 4 == 3 && n % 4 == 3)
    return -kerroin * JacobinSymboliSyt1(n, a);
  else return kerroin * JacobinSymboliSyt1(n, a);
}

// Rekursiivinen Eukleideen algoritmi.
// Laskee kahden luvun suurimman yhteisen tekijän.
static BigInteger syt(BigInteger a, BigInteger b)
{
   if (a < b) return syt(b, a);
   if (a % b == 0) return b;
   return syt(b, a % b);
}
```

Eukleideen algoritmi on tunnetusti tehokas menetelmä kahden luvun suurimman yhteisen tekijän löytämiseksi: Jokaisella Eukleideen algoritmin kierroksella, eli yhtälörivillä $b = d \cdot a + r, 0 \leq r < a$, jakojäännökset r ainakin puolittuvat lukuun b nähden, mikä tarkoittaa logaritmista määrää $\mathcal{O}(\log b)$ yhtälörivejä luvun b suhteen. Eukleideen algoritmin toteutuksessamme riittää käytännössä tehdä jokaisella rekursiokierroksella vain nopea ehtolauseen arviointi sekä yksi vaativa laskutoimitus, eli kertaluokan $\mathcal{O}(\log b \log a)$ jakojäännöslasku. Yhteensä saamme näin suoritusajaksi Eukleideen algoritmille

$$\mathcal{O}(\log b) \cdot \mathcal{O}(\log b \log a) = \mathcal{O}(\log^2 b \log a) = \mathcal{O}(\log^3 n),$$

missä b on suurempi kahdesta syöteluvusta a ja b. Teemme Jacobin symbolin laskentaa varten pienen yliarvion ja merkitsemme luvulla n suurempaa syöteluvuista a ja b.

Jacobin symbolin laskenta aiemmin kuvatulla tavalla on tehokas menetelmä: Suurimmillaan suoritusaika on, kun Jacobin symboli $\left(\frac{a}{n}\right)$ täytyy laskea useita kertoja rekursiivisesti - eli käymme läpi sanallisen algoritmin vaiheet 2-5 mahdollisimman monesti. Koska jokaisen kierroksen alussa tapahtuu luvun a vähennys modulo n, niin voisi kuvitella, että suoritusaika on suurimmillaan, kun $a = n - 1$. Näin ei kuitenkaan ole, sillä vaiheesta viisi hypättäessä takaisin ensimmäiseen vaiheeseen osoittaja a ja nimittäjä n vaihtavat paikkaansa ja tällöin osoittajan vähennys modulo n jättää osoittajaksi ykkösen, jolloin uutta rekusriokierrosta ei tarvitse suorittaa. Pisimmillään suoritusaika siis on, kun osoittaja a on noin puolet nimittäjästä n, jolloin jokaisella kierroksella laskettavat luvut ainakin puolittuvat. Pahimmassa tapauksessa on laskettava näin, kunnes $a = 1$, jolloin olemme siis enimmillään käyneet käpi logaritmisen määrän $\mathcal{O}(\log n)$ kierroksia.

Jokaisen kierroksen laskutoimitukset ovat luokkaa $\mathcal{O}(\log n)$, sillä laskutoimitusten toiset osapuolet ovat pieniä vakioita, ja niitä on enintään vakiomäärä $\mathcal{O}(1)$. Kakkosten poistaminen osoittajan a tekijöistä vaatii jokaisella kierroksella enintään $\log_2 a = \mathcal{O}(\log a) = \mathcal{O}(\log n)$ kappaletta laskutoimituksia, mutta tällöin rekursiokierrosten määrä vähenee oleellisesti. Saamme suoritusajaksi Jacobin symbolin laskemiselle siis enimmillään

$$\mathcal{O}(\log^3 n) + \mathcal{O}(\log n) \cdot ((\mathcal{O}(1) + \mathcal{O}(\log n)) \cdot \mathcal{O}(\log n)) = \mathcal{O}(\log^3 n),$$

mutta todellisuudessa tämäkin on huomattava yliarvio, johtuen muun muassa kakkosten poiston osoittajan tekijöistä rekursiokierroksia vähentävästä vaikutuksesta.

Itse Solovayn ja Strassenin testin algoritmissa teemme ennalta päätetyt k kappaletta yhtälön (3.7) pitävyyden tarkastuksia, jolloin joudumme laskemaan Jacobin symbolin sekä toistetulla neliöinnillä luvun $a^{\frac{n-1}{2}}$, joista jälkimmäisen laskutoimituksen kertaluokka on enintään $\mathcal{O}(\log^3 n)$. Vakioaikaiset toimenpiteet jättäen huomiotta saamme koko testin kertaluokaksi

$$k \cdot (\mathcal{O}(\log^3 n) + \mathcal{O}(\log^3 n)) = \mathcal{O}(k \log^3 n),$$

mikä on sama kuin Millerin ja Rabinin testin kertaluokka. Testit ovat siis suurin piirtein yhtä nopeita suorittaa, mutta Millerin ja Rabinin testi antaa hieman pienemmän todennäköisyyden virheelliselle tulokselle.

3.2.2. Prothin lause. Saamme muodostettua lauseen 3.20 seurauksena uuden, deterministisen alkulukutestin, joka perustuu Prothin lauseeseen. Prothin lause on erikoistapaus Solovayn ja Strassenin lauseen tilanteesta, eikä toimi todellakaan jokaiselle parittomalle luvulle n, mutta antaa Solovayn ja Strassenin testiä vastoin täysin varman lopputuloksen. Määrittelemme ensimmäiseksi Prothin luvut:

MÄÄRITELMÄ 3.23. Luku n on Prothin luku, jos se on muotoa $n = k \cdot 2^m + 1$, missä $m \geq 2$, $1 \leq k < 2^m$ ja k on pariton. Jos n on myös alkuluku, niin sanomme sitä Prothin alkuluvuksi.

Seuraava lause antaa oleellisimman testattavan ehdon Prothin alkulukutestille:

LAUSE 3.24. *Olkoon n Prothin luku eli $n = k \cdot 2^m + 1$, missä $m \geq 2, 1 \leq k < 2^m$ ja k on pariton. Jos on olemassa luku a siten, että $a^{\frac{n-1}{2}} \equiv -1 \mod n$, niin n on alkuluku.*

TODISTUS. Olkoon p mikä tahansa luvun n alkulukutekijä ja olkoon $d := \mathrm{ord}_p(a)$, eli luvun a kertaluku modulo p. Tässä $p \nmid a$, sillä muuten olisi $a = bp$ ja koska $p \mid n$, niin $0 \equiv (bp)^{\frac{n-1}{2}} = a^{\frac{n-1}{2}} \equiv -1 \mod p$, mikä on ristiriita. Nyt

$$a^{\frac{n-1}{2}} \equiv -1, a^{n-1} \equiv 1 \text{ ja } a^{p-1} \equiv 1 \mod p, \text{ sillä } p \nmid a$$

jolloin $d \nmid \frac{n-1}{2}$, $d \mid n-1$ ja $d \mid p-1$. Toisin sanoen $d \nmid k2^{m-1}$, $d \mid k2^m$ ja $d \mid p-1$, jolloin $2^m \mid d$ ja siten $2^m \mid p-1$. Tällöin $p = 2^m l_1 + 1$ ja koska nyt $n \equiv p \equiv 1 \mod 2^m$, niin $\frac{n}{p} \equiv 1 \mod 2^m$. Siten $n = (2^m l_1 + 1)(2^m l_2 + 1)$, missä $l_1 \geq 1$ ja $l_2 \geq 0$. Nyt kuitenkin

$$2^m l_1 l_2 < 2^m l_1 l_2 + l_1 + l_2 = \frac{n-1}{2^m} = k < 2^m,$$

joten $l_2 = 0$ ja siten $n = p$. □

Saamme vaivatta edellisten tulosten avulla todistettua, että myös edellinen väite käänteisenä on totta:

LAUSE 3.25. *Jos luku a on epäjäännös modulo n ja luku n on Prothin alkuluku, niin $a^{\frac{n-1}{2}} \equiv -1 \mod n$.*

TODISTUS. Koska n on alkuluku ja a epäjäännös modulo n, niin Legendren symboli $\left(\frac{a}{n}\right) = -1$. Lauseen 3.20 nojalla väite seuraa. □

Yhdistämällä edelliset kaksi lausetta saamme täsmällisen ehdon sille, onko Prothin luku n alkuluku:

SEURAUS 3.26 (Prothin lause). *Jos n on Prothin luku ja a on epäjäännös modulo n, niin n on alkuluku jos ja vain jos $a^{\frac{n-1}{2}} \equiv -1 \mod n$.*

Algoritmissa riittää oikeastaan etsiä vain yksi epäjäännös, jonka jälkeen totuus Prothin luvusta n selviää. Epäjäännöksen löytäminen voi kuitenkin osoittautua hyvin työlääksi. Luvulle n kuvailemme algoritmin seuraavasti:

(1) Jos n ei ole Prothin luku, niin lopetamme.
(2) Etsimme luvun $a = 2, 3, ..., n-1$, jolle $\left(\frac{a}{n}\right) = -1$. Jos $\left(\frac{a}{n}\right) = 0$, niin n on yhdistetty luku.
(3) Jos $a^{\frac{n-1}{2}} \equiv -1 \mod n$, niin n on alkuluku. Muutoin n on yhdistetty luku.

```
// Selvittää, onko luku Prothin alkuluku
static bool onAlkuluku_Proth(BigInteger n)
{
    if (!onProthinLuku(n)) throw new Exception("Syöte ei ole
       Prothin luku");
    BigInteger jacobi;
    BigInteger a;
    for (a = 3; a < n; a = a + 2)
    {
       jacobi = JacobinSymboli(a, n);
       if (jacobi == 0) return false;
       if (jacobi == -1) break;
    }

    return (BigInteger.ModPow(a, (n-1)/2, n) == n-1);
}

// Selvittää, onko luku Prothin luku
static bool onProthinLuku(BigInteger n)
{
    if (n % 2 == 0) return false;
    BigInteger erotus = n - 1;
    BigInteger kaksiPotenssiinM = 1;
    while (erotus % 2==0)
    {
       erotus = erotus / 2;
       kaksiPotenssiinM = 2 * kaksiPotenssiinM;
    }
    return (erotus < kaksiPotenssiinM);
}
```

Tässä Prothin luvuksi testaaminen vaatii enintään $1 + 3\log_2(n-1) = \mathcal{O}(\log n)$ kappaletta enintään kertaluokan $\mathcal{O}(\log_2 \log n) = \mathcal{O}(\log n)$ laskutoimituksia, joten algoritmin ensimmäinen vaihe on kertaluokaltaan $\mathcal{O}(\log^2 n)$. Seuraava vaihe on huomattavasti työläämpi: mitään (todistettuja) takeita siitä, että epäjäännös a löytyisi lukujen $2, 3, ..., n-1$ joukosta jonkin helpon kaavan mukaisesti, ei ole. Siispä kokeilemme tässä tapauksessa kaikki luvut a pienimmästä alkaen läpi, kunnes löydämme epäjäännöksen. Tämä voi tapahtua erittäin nopeasti, tai sitten ei.

Äärimmäisessä tapauksessa siis on testattava huomattava määrä eri lukuja a. Pienin epäjäännös modulo n paitsi ykköstä suurempi myös alkuluku: jos olisi yhdistetty luku $m = ab$ pienin epäjäännös modulo n, niin tekijät a ja b olisivat neliöjäännöksiä, jolloin myös m olisi neliöjäännös. Epäjäännöksiksi siis riittäisi testata vain alkulukuja, mutta koska alkuluvun tunnistaminen vie runsaasti aikaa, tyydymme tarkastelemaan vain parittomia lukuja kakkosen lisäksi. Tällöin siis riittää testata enintään $\frac{n}{2}$ lukua a, jotta löydämme epäjäännöksen. Näillä tiedoilla epäjäännöksen löytäminen

on kuitenkin kertaluokan

$$(\frac{n}{2}+1)\cdot\mathcal{O}(\log^3 n)=\mathcal{O}(n\log^3 n)$$

tehtävä, vaikka käytännössä epäjäännös löytyy huomattavasti helpommin.

Algoritmin viimeinen vaihe vaatii vain yhden toistetun neliöinnin kertaluokkaa $\mathcal{O}(\log^3 n)$, joten yhteenlaskemalla edellisten vaiheiden kertaluokat saamme varsinaisen algoritmin kertaluokaksi $\mathcal{O}(n\log^3 n)$. Keskimäärin algoritmin suoritus on kuitenkin lähempänä kertaluokkaa $\mathcal{O}(k\log^3 n)$, missä k on huomattavasti lukua n pienempi luku: etsimällä pienintä epäjäännöstä miljoonalle enimmäiselle luonnolliselle luvulle voimme huomata, että suurin näistä pienimmistä epäjäännöksistä on 71, tyypillisemmin jopa alle 10. Käytännössä kyseessä on siis varsin tehokas deterministinen testi.

3.3. Lucasin ja Lehmerin testi

Lucasin ja Lehmerin testi on deterministinen ja hyvin yksinkertainen alkulukutesti, joka toimii ainoastaan *Mersennen alkuluvuille* $M_p=2^p-1$, missä p on alkuluku. Se on myös helposti toteutettavissa tietokoneella. Kuten voimme Mersennen alkulukujen yleisestä muodosta päätellä, voi testiä käyttää hyvin suurten alkulukujen etsimiseen. Kuitenkaan Mersennen alkulukuihin rajoittumisensa vuoksi Lucasin ja Lehmerin testistä ei juuri muuta iloa ole.

Mikäli tavoitteenamme on siis etsiä hyvin suuria alkulukuja, kuten 22 338 618-numeroinen ja toistaiseksi (tammikuussa 2016) suurin löydetty alkuluku

$$M_{74207281}=2^{74207281}-1,$$

niin Lucasin ja Lehmerin testi on siihen hyödyllisin; RSA-salaukseen se on käyttökelvoton, sillä Mersennen alkulukuja ei ole olemassa kovin montaa käytännöllisessä kokoluokassa. Mersennen alkulukujen välissä on toki muitakin alkulukuja; tämän tiedämme esimerkiksi Bertrandin postulaatista, jonka mukaan kaikille $n>1$ lukujen n ja $2n$ välissä on aina ainakin yksi alkuluku [**9**, s. 343].

Lucasin ja Lehmerin testi perustuu seuraavaan lauseeseen:

Lause 3.27 (Lucasin ja Lehmerin lause). *Olkoon p alkuluku ja jono (S_n) määritelty rekursiivisesti*

$$\begin{cases} S_1 &= 4 \\ S_n &= S_{n-1}^2-2 \end{cases}$$

Jos luku $M_p=2^p-1$ jakaa luvun S_{p-1}, niin M_p on alkuluku.

Tämän tuloksen todistamiseksi tarvitsemme muutaman lemman:

Lemma 3.28. *Olkoon $\omega=2+\sqrt{3}$ ja $\overline{\omega}=2-\sqrt{3}$. Tällöin $S_n=\omega^{2^{n-1}}+\overline{\omega}^{2^{n-1}}$.*

Todistus. Helposti voimme huomata, että $\omega\overline{\omega}=1$. Todistamme lemman induktiolla luvun n suhteen.

Selvästi $\omega^{2^{1-1}}+\overline{\omega}^{2^{1-1}} = 4 = S_1$ ja $\omega^{2^{2-1}}+\overline{\omega}^{2^{2-1}} = (7+2\sqrt{3})+(7-2\sqrt{3}) = 14 = S_2$. Oletamme, että yhtälö $S_k = \omega^{2^{k-1}} + \overline{\omega}^{2^{k-1}}$ pätee luvuille $k \leq 2$. Tällöin

$$\begin{aligned}
\omega^{2^{(k+1)-1}} + \overline{\omega}^{2^{(k+1)-1}} &= \omega^{2^{(k-1)+1}} + \overline{\omega}^{2^{(k-1)+1}} \\
&= (\omega^{2^{k-1}})^2 + (\overline{\omega}^{2^{k-1}})^2 \\
&= (\omega^{2^{k-1}})^2 + (\overline{\omega}^{2^{k-1}})^2 + 2(\omega\overline{\omega})^{2^{k-1}} - 2(\omega\overline{\omega})^{2^{k-1}} \\
&= (\omega^{2^{k-1}} + \overline{\omega}^{2^{k-1}})^2 - 2 \\
&= S_k^2 - 2,
\end{aligned}$$

jolloin olemme todistaneet induktioväitteen. □

Jos siis luku M_p jakaa luvun S_{p-1}, niin $\omega^{2^{p-2}} + \overline{\omega}^{2^{p-2}} = cM_p$, missä c on kokonaisluku. Tällöin kertomalla yhtälön molemmat puolet luvulla $\omega^{2^{p-2}}$ saamme tuloksen

$$\omega^{2^{p-1}} = cM_p\omega^{2^{p-2}} - 1. \tag{3.9}$$

Neliöimällä tämän saamme yhtälön

$$\omega^{2^p} = \left(cM_p\omega^{2^{p-2}} - 1\right)^2. \tag{3.10}$$

Todistaaksemme väitteen muodostamme antiteesin: oletamme, että M_p on yhdistetty luku ja valitsemme sille alkulukujakajan $q > 2$, jolle $q^2 \leq M_p$.

Lucasin ja Lehmerin lauseen todistus. Olkoon $\mathbb{Z}_q$ jäännösluokkien joukko modulo q ja $X = \{a + b\sqrt{3} : a, b \in \mathbb{Z}_q\}$ sekä muunnellut yhteen- ja kertolaskut laskutoimituksia joukossa X. Voimme siis ajatella joukon X koostuvan ekvivalenssiluokkapareista, kuten voimme ajatella kompleksilukujen koostuvan reaalilukupareista. Määrittelemme alkioiden $x = (x_1 + x_2\sqrt{3}), y = (y_1 + y_2\sqrt{3}) \in X$ yhteenlaskun

$$x + y = (x_1 + x_2\sqrt{3}) + (y_1 + y_2\sqrt{3}) = (x_1 + y_1) + (x_2 + y_2)\sqrt{3}$$

ja kertolaskun

$$xy = (x_1 + x_2\sqrt{3})(y_1 + y_2\sqrt{3}) = (x_1y_1 + 3x_2y_2) + (x_1y_2 + x_2y_1)\sqrt{3}.$$

Kertoimien väliset laskutoimitukset toimivat kuten tavallisesti ekvivalenssiluokkien yhteen- ja kertolasku. Näin määrittelemällä yhteen- ja kertolasku – eli kuvaukset $+, \cdot : X \times X \to X$ – ovat todellakin laskutoimituksia joukossa X, sillä laskutoimitusten tulokset pysyvät joukossa X. Helposti näemme, että näillä laskutoimituksilla varustettuna X on rengas.

Olkoon X^* kertolaskun suhteen kääntyvien alkioiden joukko, jolloin X^* on ryhmä. Lagrangen lauseen nojalla jokaisen ryhmän X^* alkion kertaluku on enintään $q^2 - 1$, sillä ainakaan 0 ei ole kääntyvä alkio.

Oletimme jo aiemmin vastoin väitettä, että on olemassa luku $q \mid M_p,\ q^2 \leq M_p$. Nyt voimme tulkita, että $\omega \in X$ ja koska q jakaa luvun M_p, niin $cM_p\omega^{2^{p-2}} = 0$ renkaan X alkiona. Renkaassa X yhtälön (3.10) seurauksena saamme

$$\omega^{2^p} = 1,$$

jolloin $\omega \in X^*$ on kääntyvä. Lagrangen lauseen ja yhtälön (3.10) nojalla $\operatorname{ord}\omega \mid 2^p$, jolloin $\operatorname{ord}\omega$ on kakkosen potenssi. Yhtälön (3.9) ja ehdon $cM_p\omega^{2^{p-2}} = 0$ nojalla

$\operatorname{ord}\omega \not| 2^{p-1}$. Nyt siis $\operatorname{ord}\omega = 2^p$.

Nyt Lagrangen lauseen perusteella $2^p \leq q^2 - 1$, mutta toisaalta $q^2 - 1 \leq M_p - 1 = 2^p - 2$, mikä on ristiriita. □

Nyt olemme onnistuneet muotoilemaan hyvin käytännöllisen tuloksen suurten alkulukujen löytämistä varten. Lausetta hyödyntäessämme joudumme kuitenkin käsittelemään melko suuria lukuja S_{p-1} ja $M_p = 2^p - 1$: esimerkiksi, kun testaamme alkuluvulle $p = 9689$, onko luku M_{9689} alkuluku, on jo pelkästään M_{9689} pituudeltaan 2917 numeroa kymmenjärjestelmässä. Lisäksi luku S_{p-1} on vielä tätäkin suurempi.

Lucasin ja Lehmerin lauseen ehdon $M_p \mid S_{p-1}$ pitävyyden osoittamiseksi riittää näyttää, että on voimassa

$$S_{p-1} \equiv 0 \mod M_p.$$

Tällöin voimme laskea luvun $S = S_{p-1} \mod M_p$ asettamalla aloitusehdoksi $S = 4$ ja tämän jälkeen laskemalla $p - 2$ kertaa $S := S^2 - 2 \mod M_p$. Algoritmiksi saamme ohjelmakoodin muodossa kaikessa yksinkertaisuudessaan seuraavaa:

```
// Laskee Lucasin ja Lehmerin testiä käyttäen, onko luku 2^p-1
// alkuluku. Syötteen p on oltava alkuluku.
static bool onAlkuluku_LucasLehmer(int p)
{
    BigInteger S = 4;
    BigInteger Mp = BigInteger.Pow(2, p) - 1;
    for (int i = 1; i < p - 1; i++)
    {
        S = ((S * S) - 2) % Mp;
    }
    return (S == 0);
}
```

Algoritmi siis palauttaa arvon tosi, jos $S_{p-1} \equiv 0 \mod M_p$ ja epätosi muutoin. Algoritmi on hyvin nopea ja laskutoimitusten määrä kasvaa lineaarisesti syötteen p kasvaessa, mutta luvulla $M_p = 2^p - 1$ laskeminen teettää työtä. Binäärilukuna Mersennen alkuluvut ovat kuitenkin helppoja muodostaa: niissä on vain p kappaletta ykkösiä, tai heksadesimaalina $\left\lfloor \frac{p}{4} \right\rfloor$ kappaletta numeroita `F` ja etummaisin numero on jakojäännöksen $p \mod 4$ mukaan jokin numeroista `1`, `3` tai `7`. Tämän vuoksi luvun M_p ”laskemiseen” ei tarvitse juurikaan tuhlata aikaa, eikä sen määrittäminen muutenkaan muodosta merkittävää osaa algoritmin seuraavien askelten eli lukujonon jäsenten määrittämisen rinnalla.

Lukujonon (S_n) jäsenen S_{p-1} laskemiseen käymme läpi $p-2$ vaihetta, joissa jokaisessa on yksi luokan $\mathcal{O}(\log^2(2^p - 1)) = \mathcal{O}(p^2)$ kertolasku, $\mathcal{O}(p^2)$-vähennyslasku sekä luokan $\mathcal{O}(\log(2^{2p}) \log(2^p - 1)) = \mathcal{O}(p^2)$ jakojäännöslasku. Laskutoimitusten lukumääräksi saamme siis $(p-2) \cdot (3 \cdot \mathcal{O}(p^2)) = \mathcal{O}(p^3)$. Koska testattava luku $n = 2^p - 1$, niin

luvun n kokoon nähden algoritmi on kertaluokaltaan

$$\mathcal{O}(p^3) = \mathcal{O}(\log_2^3(n+1)) = \mathcal{O}(\log^3 n).$$

Lucasin ja Lehmerin testi on siis paitsi deterministinen, mutta suoritusajaltaan myös nopeampi kuin Millerin ja Rabinin. Testi kuitenkin toimii hyvin rajoitetulle määrälle kokonaislukuja, ja lisäksi se on suoritettava aina loppuun asti; esimerkiksi Millerin Rabinin testissä pystyimme lopettamaan testin mahdollisesti hyvin aikaisessa vaiheessa ja toteamaan testatun luvun yhdistetyksi, mutta Lucasin ja Lehmerin algoritmin suoritus vie saman verran aikaa, oli testattava luku alkuluku tai ei.

LUKU 4

AKS-alkulukutesti

Historiallisesti alkulukutestauksen algoritmien kehityksessä ei varsinaisesti tapahtunut suuria harppauksia pariin vuosikymmeneen ennen vuotta 2002, jolloin intialaiset tietoteknikot Manindra Agrawal, Neeraj Kayal ja Nitin Saxena julkaisivat uuden alkulukutestin. Testin käsittelemiseen tarvitsemme runsaasti polynomirenkaita koskevaa algebran teoriaa. Seuraava kappale on tarkoitettu lähinnä pikaiseksi kertaukseksi, mutta sisältää pari tuntemattomampaa tulosta liittyen tulevan alkulukutestin toimivuuden osoittamiseen.

4.1. $\mathbb{Z}_n$-kertoimiset polynomirenkaat

Ensimmäisenä käsittelemme merkintöjä: $\mathbb{Z}_n[X]$ on niiden polynomien joukko, joiden kertoimet ovat renkaan $\mathbb{Z}_n$ alkioita. Polynomissa merkitsemme ainoaa muuttujaa symbolilla X. Määrittelemällä polynomien yhteen- ja kertolaskut tavanomaisesti on myös $\mathbb{Z}_n[X]$ rengas.

Polynomin $P(X) = \sum_{k=0}^{m} a_k X^k \in \mathbb{Z}_n[X]$ määrämä polynomifunktio on

$$P : \mathbb{Z}_n \to \mathbb{Z}_n, P(x) = \sum_{k=0}^{m} a_k x^k.$$

Sanomme, että alkio $\omega \in \mathbb{Z}_n$ on polynomin P juuri, jos $P(\omega) = 0$.

Polynomien kertoimet voivat toki olla abstraktimmasta joukosta. Olkoon esimerkiksi K jokin kommutatiivinen rengas. Määrittelemme K-kertoimisten polynomien (polynomi)kuvaukset myös varsin luonnollisella tavalla: jos $P(X) = \sum_{k=0}^{m} a_k X^k \in K[X]$, niin polynomia P vastaava kuvaus on

$$P : K[X] \to K[X], P(R(X)) = \sum_{k=0}^{m} a_k R(X)^k,$$

missä polynomien yhteen- ja kertolaskut edelleen ovat tutulla tavalla määriteltyjä.

Voimme määritellä jaollisuuden polynomeille kuten kokonaisluvuille.

MÄÄRITELMÄ 4.1. Olkoot polynomit $P, Q \in \mathbb{Z}_n[X]$. Sanomme, että polynomi Q on jaollinen polynomilla P, jos on olemassa polynomi $R \in \mathbb{Z}_n[X]$ siten, että $Q(X) = P(X)R(X)$. Merkitsemme tällöin, että $P(X) \mid Q(X)$.

Kuten reaalipolynomien puolelta voimme muistaa, niin $\omega \in \mathbb{Z}_n$ on polynomin $P \in \mathbb{Z}_n[X]$ juuri täsmälleen silloin, kun $(X - \omega) \mid P(X)$ [**16**, s. 81]. Lisäksi, jos $\mathbb{Z}_n$ on

kunta eli n on alkuluku, niin polynomilla P juuria on olemassa korkeintaan asteluvun $\deg(P)$ verran [**5**, s. 111]. Nämä tulokset pätevät myös hieman yleisemmillekin polynomeille, joiden kertoimet ovat esimerkiksi joissain aivan muissa sopivissa renkaissa tai kunnissa kuin nyt käsittelemässämme $\mathbb{Z}_n$-renkaassa. Tästä tiedosta on hyötyä erityisesti, kun p on alkuluku ja h jaoton polynomi ja käsittelemme tällöin muodostuvan kunnan $F = \mathbb{Z}_p[X]/h$ määräämiä polynomeja, eli renkaan $F[X]$ alkioita.

Osoittautuu, että kuten kokonaisluvut, myös kuntakertoimiset polynomit ovat ilmoitettavissa yksikäsitteisesti järjestystä lukuun ottamatta jaottomien, pienempiasteisten polynomien tulona. Tästä on hyötyä erityisesti silloin, kun käsittelemme $\mathbb{Z}_p$-kertoimisia polynomirenkaita, missä p on alkuluku.

LAUSE 4.2. *Olkoon F kunta. Tällöin jokainen polynomi $f \in F[X], f \neq 0$ on esitettävissä yksikäsitteisesti tulona $a \cdot h_1 \cdots h_s, s \geq 0$, missä kerroin $a \neq 0 \in F$ ja polynomit $h_1, ..., h_s \in F[X]$ ovat jaottomia, korkeimman asteen termin kertoimeltaan 1 ja asteluvultaan suurempia kuin 0. Esitys on yksikäsitteinen tekijäpolynomien järjestystä lukuun ottamatta.*

TODISTUS. Sivuutamme todistuksen tässä tutkielmassa; ks. [**5**, s. 109]. □

Jaollisuuden määritelmän avulla myös polynomien kongruenssiyhtälöiden määrittely on varsin helppoa. Kongruenssit ovat kenties oleellisin apuväline, jota hyödynnämme AKS-testissä.

MÄÄRITELMÄ 4.3. Polynomit $P, Q \in \mathbb{Z}_n[X]$ ovat kongruentteja modulo $R(X)$, jos $R(X) \mid P(X) - Q(X)$. Merkitsemme tällöin $P(X) \equiv Q(X) \mod R(X)$.

Seuraavia lauseita tarvitsemme AKS-alkulukutestin toimivuuden osoittamisessa:

LEMMA 4.4. *Olkoon polynomi $h \in \mathbb{Z}_n[X], h \neq 0$ sellainen, että sen korkeimman asteen termin kerroin on $1 \in \mathbb{Z}_n$ ja polynomit f_1, f_2, g_1, g_2 siten, että $f_1 \equiv f_2$ ja $g_1 \equiv g_2 \mod h$. Tällöin*

(a) $f_1 + g_1 \equiv f_2 + g_2 \mod h$
(b) $f_1 \cdot g_1 \equiv f_2 \cdot g_2 \mod h$
(c) $f(g_1) \equiv f(g_2) \mod h$ *kaikille polynomeille* $f \in \mathbb{Z}_n[X]$.

TODISTUS. Oletamme, että $f_1 - f_2 = hq_1$ ja $g_1 - g_2 = hq_2$. Tällöin

$$(f_1 + g_1) - (f_2 + g_2) = h(q_1 + q_2)$$

ja

$$f_1 \cdot g_1 - f_2 \cdot g_2 = (f_1 - f_2)g_1 + f_2(g_1 - g_2) = h(g_1 q_1 + f_2 q_2).$$

Jos polynomi f on vakiopolynomi tai $f = X$, niin (c)-kohdassa ei ole todistettavaa. Käyttämällä (b)-kohtaa toistuvasti saamme todistettua väitteen kaikille monomeille $f = aX^k$, missä $a \in \mathbb{Z}_n$ ja $k \geq 0$. Tämän tiedon ja (a)-kohdan avulla käsittelemällä polynomeja monomien summina saamme väitteen koskemaan kaikkia polynomeja. □

LAUSE 4.5. *Jos p on alkuluku, niin*

(a) $(f + g)^p = f^p + g^p$ *ja* $(f \cdot g)^p = f^p \cdot g^p$ *kaikille polynomeille* $f, g \in \mathbb{Z}_p[X]$.
(b) *Jokaiselle polynomille* $f \in \mathbb{Z}_p[X]$ *ovat voimassa yhtälöt* $f^p = f(X^p)$ *ja yleisemmin* $f^{p^k} = f(X^{p^k})$ *kaikille* $k \geq 0$.

TODISTUS. Saamme yhtälön $(f \cdot g)^p = f^p \cdot g^p$ seurauksena renkaan $\mathbb{Z}_\mathrm{p}$ polynomien kertolaskun kommutatiivisuudesta, joka on puolestaan suora seuraus renkaan $\mathbb{Z}_\mathrm{p}$ kertolaskun kommutatiivisuudesta. Binomikaavan nojalla voimme polynomien summan potenssin osalta kirjoittaa

$$(f+g)^p = f^p + \sum_{i=1}^{p-1} \binom{p}{i} f^i \cdot g^{p-i} + g^p.$$

Tässä summan binomikertoimet $\binom{p}{i}$ voimme kirjoittaa kaikille $1 \leq i \leq p-1$ muodossa

$$\binom{p}{i} = \frac{p(p-1)\cdots(p-i+1)}{i(i-1)\cdots 2 \cdot 1},$$

missä osoittaja on jaollinen on jaollinen luvulla p, mutta koska p on alkuluku, niin kaikki nimittäjän tekijät $1, 2, ..., i$ ovat suhteellisia alkulukuja luvulle p ja siten nimittäjä ei ole jaollinen luvulla p. Näin ollen $\binom{p}{i} \equiv 0 \mod p$ kaikille $1 \leq i \leq p-1$ ja siten summa $\sum_{i=1}^{p-1} \binom{p}{i} f^i \cdot g^{p-i} = 0$ renkaassa $\mathbb{Z}_\mathrm{p}[X]$ ja $(f+g)^p = f^p + g^p$.

Olkoon $f = a_k X^k + ... + a_1 X + a_0$. Soveltamalla $k-1$ kertaa (a)-kohdan sääntöä $(f+g)^p = f^p + g^p$ voimme päätellä polynomipotenssin f^p olevan sama kuin

$$\begin{aligned} f^p &= a_k^p (X^k)^p + ... + a_1^p X^p + a_0^p \\ &= a_k (X^p)^k + ... + a_1 (X^p) + a_0 \text{ Fermat'n pienen lauseen nojalla.} \end{aligned}$$

Yleisemmän tapauksen saamme osoitettua induktiolla. Tapauksessa $k = 0$ ei ole todistettavaa, sillä $f = f(X)$ ja tapauksen $k = 1$ käsittelimme edellä. Oletamme, että $k \geq 2$ ja että väite pätee kaikille indekseille $0, 1, ..., k-1$. Tällöin

$$f^{p^k} = (f^{p^{k-1}})^p \underset{(1)}{=} (f(X^{p^{k-1}}))^p \underset{(2)}{=} f((X^p)^{p^{k-1}}) = f(X^{p^k}),$$

missä yhtäsuuruudet (1) ja (2) seuraavat molemmat induktio-oletuksesta. □

4.2. Perusteet deterministiselle polynomiaikaiselle alkulukutestille

AKS-alkulukutesti perustuu seuraavaan polynomirenkaita koskevaan lemmaan, joka on yleistys Fermat'n pienestä lauseesta.

LAUSE 4.6. *Olkoot luvut $a, n \in N, n \geq 2, n > a$ siten, että* $\mathrm{syt}(a, n) = 1$. *Tällöin luku n on alkuluku jos ja vain jos renkaassa $\mathbb{Z}_\mathrm{n}[X]$ on voimassa*

$$(X+a)^n = X^n + a. \tag{4.1}$$

TODISTUS. Tunnetusti voimme kirjoittaa polynomin $(X+a)^n$ binomikaavan mukaan summana

$$(X+a)^n = X^n + \sum_{i=1}^{n-1} \binom{n}{i} a^i X^{n-i} + a^n.$$

Väitteen toinen suunta seuraa suoraan lauseesta 4.5. Todistaaksemme väitteen toisen puolen oletamme, että n ei ole alkuluku. Tällöin luvulla n on oltava alkulukutekijä

$p < n$ ja jollekin $k \geq 1$ luku p^k jakaa luvun n, mutta $p^{k+1} \nmid n$. Nyt binomikaavasta polynomille $(X + a)^n$ saamamme X^{n-p}-termin kerroin on

$$\binom{n}{p} \cdot a^p = \frac{n(n-1)\cdots(n-p+1)}{p!} \cdot a^p,$$

jonka osoittajassa ainoastaan kerroin n on jaollinen luvulla p^k ja muut kertoimet ovat suhteellisia alkulukuja luvulle p. Tällöin osoittaja on jaollinen luvulla p^k, muttei luvulla p^{k+1}. Nimittäjä $p!$ on jaollinen luvulla p. Koska a ja n ovat oletuksen nojalla suhteellisia alkulukuja, niin $p \nmid a^p$ ja siten $\binom{n}{p} \cdot a^p$ ei voi olla jaollinen luvulla p^k eikä tällöin myöskään luvulla n. Nyt siis $\binom{n}{p} \cdot a^p \neq 0$ renkaassa $\mathbb{Z}_\mathrm{n}$ ja siten $(X+a)^n \neq X^n + a$ polynomirenkaassa $\mathbb{Z}_\mathrm{n}[X]$.

□

Lauseen 4.6 asetelma voisi nopeasti katsottuna vaikuttaa alkulukutestiltä, mutta tarkemmin tarkasteltuna lause on hyödytön sellaisenaan testaamiseen: testattavaa lukua n kohden saamme n-asteisen polynomin, josta on tutkittava n kappaletta X^i-termien kertoimia $\binom{n}{i} a^{n-i}, 0 < i < n$. Binomikaavan avulla polynomin $(X + a)^n$ auki laskeminen olisi hyvin työlästä, sillä jo yksittäisen binomikertoimen arviointi suurelle luvulle n vaatii runsaasti laskutoimituksia.

Polynomin $(X + a)^n$ kertoimien laskemiseen voimme kuitenkin hyödyntää toisenlaista laskutapaa: kuten aiemmin laskimme toistettua neliöintiä käyttäen lukuja a^n, voimme tehdä saman myös polynomeille $(X + a)^n$. Jos n on parillinen, niin hajautamme polynomin neliöksi $((X + a)^{n/2})^2$, ja luvun n ollessa pariton tuloksi $(X + a)((X + a)^{\frac{n-1}{2}})^2$. Toistamalla tätä rekursiivisesti saadun uuden polynomitulon $(X + a)^k$ laskemiseksi riittää tehdä luvun n suhteen logaritminen määrä $\mathcal{O}(\log n)$ polynomien kertolaskuja. Vastaavasti kuin toistetussa neliöinnissä saamme pahimmassa tapauksessa, eli jokaisella kierroksella eksponentin ollessa pariton, polynomien kertolaskuja laskettavaksi kaksi kappaletta kierrosta kohden. Polynomien asteluvut voivat kuitenkin olla edelleen suuria ja p- ja q-asteisten polynomien kertolaskun vaatiessa noin $\mathcal{O}(pq)$ laskutoimitusta on selvää, että jotain uutta on vielä keksittävä, jottei kertolaskuja tarvitse tehdä kohtuutonta määrää.

AKS-testin luojat kehittivät oikotien yhtälön (4.1) arvioimiseen: Polynomirenkaassa $\mathbb{Z}_\mathrm{n}[X]$ kahden polynomin P ja Q ollessa samoja eli $P = Q$, ovat ne samoja myös modulo kolmas polynomi R eli $P \equiv Q \mod R$. Jos valitsemmme polynomiksi $Q(X) = X^r - 1$, missä r on sopivasti valittu ja kooltaan logaritminen lukuun n nähden, niin saamme radikaalisti pudotettua polynomien toistetun neliöinnin laskutoimitusten lukumäärää. Voimme muuttaa yhtälön (4.1) muotoon

$$(X + a)^n \equiv X^n + a \mod (X^r - 1), \tag{4.2}$$

jolloin riittää verrata jäännöspolynomien

$$(X + a)^n \mod (X^r - 1) \text{ ja } X^{n \bmod r} + a$$

kertoimia; tässä saamme jälkimmäisen polynomin $X^{n \bmod r} + a$ siitä, että $X^n + a \equiv X^{n \bmod r} \cdot (X^r)^k + a \equiv X^{n \bmod r} \cdot (1)^k + a = X^{n \bmod r} + a \mod (X^r - 1)$ jollekin

luvulle k, jolle $n = kr + (n \mod r)$. Laskiessamme auki polynomia $(X + a)^n$ toistettua neliöintiä käyttäen voimme nyt laskea osatuloksia modulo $(X^r - 1)$, jolloin laskutoimitusten lukumäärä pysyy polynomien kertolaskuissakin kohtuullisena astelukujen ollessa logaritmisia lukuun n nähden.

Yhtälö (4.2) on siis voimassa, jos n on alkuluku, mutta toiseen suuntaan ehdon pitävyyden selvittämiseen tarvitsemme lisätoimia: voisihan olla niin, että jollekin yhdistetylle luvulle n yhtälö (4.2) on voimassa, mutta alkuperäinen (4.1) ei ole. Osoittautuu, että kun löydämme sopivan luvun r, niin riittää testata yhtälön (4.2) pitävyyttä sarjalla lukuja $a = 1, 2, ..., \psi(r)$, missä $\psi(r)$ on sopivan pieni luvusta r riippuva luku.

4.3. AKS-algoritmi ja sen toimivuus

Varsinainen AKS-algoritmi toimii seuraavasti:

Algoritmi luvulle $n > 1$.

(1) Jos $n = a^b$ joillekin luvuille $a, b > 1$, niin n on yhdistetty luku.
(2) Etsimme pienimmän luvun r siten, että $\mathrm{ord}_r(n) > (\log_2 n)^2$.
(3) Jos $1 < \mathrm{syt}(a, n) < n$ jollekin luvuista $2 \leq a \leq r$, niin n on yhdistetty luku.
(4) Jos $n \leq r$, niin n on alkuluku.
(5) Jos kaikille $1 \leq a \leq \lfloor \sqrt{\varphi(r)} \log_2 n \rfloor$ on renkaassa $\mathbb{Z}_n[X]$ voimassa yhtälö $(X+a)^n \equiv X^n+a \mod (X^r-1)$, niin n on alkuluku. Muutoin n on yhdistetty luku.

Jotta algoritmissa olisi mitään järkeä, on syytä näyttää sopivan luvun r olemassaolo. Jos tällainen luku r on olemassa, niin on selvää, että jos testattava luku n on alkuluku, niin algoritmi palauttaa vastaukseksi ”alkuluku”. Toiseen suuntaan tilanne ei ole aivan yhtä selvä: jos algoritmi palauttaa vastaukseksi ”alkuluku”, niin onko luku n todella alkuluku? Osoittautuu, että näin todella on, mutta se vaatii useita aputuloksia.

Lause 4.7. *Luku n on alkuluku jos ja vain jos AKS-algoritmi palauttaa vastaukseksi ”alkuluku”.*

Jos luku n on alkuluku, niin selvästikään se ei ole minkään luvun potenssi ja se on suhteellinen alkuluku kaikille luvuille, paitsi omille monikerroilleen, joten se läpäisee vaiheet (1) ja (3). Jos vaiheessa (4) luku $r \geq n$, niin koska luku n on vaiheen (3) läpäisyn nojalla suhteellinen alkuluku kaikille itseään pienemmille luvuille, palauttaa algoritmi oikean vastauksen myös tässä kohtaa.

Luvun n ollessa alkuluku algoritmin vaiheen (5) ehdot toteutuvat, sillä renkaassa $\mathbb{Z}_p[X]$ yhtälö $(X + a)^n \equiv X^n + a \mod (X^r - 1)$ toteutuu lauseen 4.6 ja tämän jälkeisten päättelyidemme nojalla.

Toiseen suuntaan väite ei ole aivan niin selvä. Käytännön syistä luvun r on oltava kooltaan logaritminen lukuun n nähden, sillä muuten vaihe (5) ei ole suoritettavissa polynomisessa ajassa eikä testin käsittelemisessä olisi juuri mitään mieltä. Ensin näytämme, että on olemassa sopiva luku r, joka toteuttaa vaiheen (2) ehdon $\mathrm{ord}_r(n) > (\log_2 n)^2$.

LEMMA 4.8 (Nair'n lemma). *Lukujen $1, 2, ..., n$, missä $n \geq 7$, pienin yhteinen jaettava on vähintään 2^n.*

TODISTUS. Ks. [**14**, ss. 126-127] □

LAUSE 4.9. *On olemassa luku $r \leq \max\{3, \lceil(\log_2 n)^5\rceil\}$ siten, että $\mathrm{syt}(r, n) = 1$ ja $\mathrm{ord}_r(n) > (\log_2 n)^2$.*

TODISTUS. Luvuille $n = 2, 3, 4$ sopivat luvut ovat $r = 3, 5, 11$, joten voimme olettaa, että $n \geq 5$. Tällöin $B := \lceil(\log_2 n)^5\rceil \geq 68$, jolloin lemman 4.8 mukaan lukujen $1, 2, ..., B$ pienin yhteinen jaettava on ainakin 2^B.

Olkoon nyt luku $r \leq B$ pienin sellainen luku, joka ei jaa tuloa

$$P := n^{\lfloor \log_2 B\rfloor} \cdot \prod_{i=1}^{\lfloor(\log_2 n)^2\rfloor} (n^i - 1).$$

Tällainen luku r on olemassa: Jos ei olisi, niin kaikille luvuille $d = 1, 2, ..., B$ luku d jakaisi tulon P, jolloin lukujen $1, 2, ..., B$ pienin yhteinen jaettava jakaisi myös tulon P. Toisaalta

$$\begin{aligned}
P &< n^{\lfloor \log_2 B\rfloor} \cdot \textstyle\prod_{i=1}^{\lfloor(\log_2 n)^2\rfloor} n^i \\
&= n^{\lfloor \log_2 B\rfloor + \sum_{i=1}^{\lfloor(\log_2 n)^2\rfloor} i} \\
&= n^{\lfloor \log_2 B\rfloor + \frac{1}{2}\lfloor(\log_2 n)^2\rfloor(\lfloor(\log_2 n)^2\rfloor + 1)} \\
&\leq n^{5\log_2(\log_2 n + 1) + \frac{1}{2}(\log_2 n)^4 + \frac{1}{2}(\log_2 n)^2} \\
&\underset{(1)}{\leq} n^{(\log_2 n)^4} \\
&= 2^{(\log_2 n)(\log_2 n)^4} \\
&= 2^{(\log_2 n)^5} \\
&\leq 2^{\lceil(\log_2 n)^5\rceil} \\
&= 2^B.
\end{aligned}$$

Helposti näemme, että epäyhtälö (1) pätee, kun $n \geq 5$. Kuitenkin lukujen $1, 2, ..., B$ pienin yhteinen jaettava jakaa luvun P ja on vähintään 2^B Nair'n lemman 4.8 nojalla. Päädyimme siis ristiriitaan ja siten luvun r on oltava olemassa.

Helposti voimme huomata, että $\max\{k \in \mathbb{N} : q^k \mid r \text{ ja } q \text{ on alkuluku}\} \leq \lfloor \log_2 B\rfloor$, sillä muutoin olisi oltava $r > B$. Tällöin

$$r \mid \prod_{\substack{q \mid r \\ q \text{ on alkuluku}}} q^{\lfloor \log_2 B\rfloor}.$$

Nyt luku n ei voi olla jaollinen kaikilla luvun r alkulukutekijöillä, sillä muuten

$$\prod_{\substack{q \mid r \\ q \text{ on alkuluku}}} q^{\lfloor \log_2 B \rfloor} \mid n^{\lfloor \log_2 B \rfloor},$$

mikä tarkoittaisi sitä, että $r \mid P$, mutta tämä ei ole mahdollista luvun r valinnan nojalla. Erityisesti nyt $\mathrm{syt}(r,n) < r$ ja $\frac{r}{\mathrm{syt}(r,n)} > 1$. Lisäksi luku P ei ole jaollinen luvulla $\frac{r}{\mathrm{syt}(r,n)}$: Jos olisi, niin voisimme merkitä luvun r alkulukupotenssien tulona $r = \prod_{q \mid r} q^{k_q}$, missä jokaiselle $q \nmid n$ pätisi $q^{k_q} \mid \prod_{i=1}^{\lfloor (\log_2 n)^2 \rfloor}(n^i - 1)$. Lisäksi jokaiselle luvun r alkulukutekijälle $q \mid n$ puolestaan $k_q \leq \lfloor \log_2 B \rfloor$, joten $q^{k_q} \mid n^{\lfloor \log_2 B \rfloor}$, ja näiden päätelmiemme seurauksena $r \mid P$, mikä on jälleen ristiriita.

Nyt siis $1 < \frac{r}{\mathrm{syt}(r,n)} \leq r$, mutta koska luku r oli pienin niistä luvuista, joka ei jaa lukua P, niin on oltava $r = \frac{r}{\mathrm{syt}(r,n)}$ eli $\mathrm{syt}(r,n) = 1$.

Koska kertaluvun modulo r määritelmän mukaan $n^{\mathrm{ord}_r(n)} \equiv 1 \mod r$, niin

$$r \mid n^{\mathrm{ord}_r(n)} - 1.$$

Koska $r \nmid P$, niin luku r ei jaa mitään luvuista $(n^i - 1)$, missä $1 \leq i \leq \lfloor (\log_2 n)^2 \rfloor$. Siten on oltava $\mathrm{ord}_r(n) > (\log_2 n)^2$. □

Luvun r olemassaolon myötä voimme jo vähitellen todeta, että algoritmissa saattaa olla jotain järkeä. Oletamme nyt viimeistään, että käymme AKS-algoritmin läpi jollain luvulla n, ja algoritmi antaa vastaukseksi ”alkuluku”. Lauseen 4.9 nojalla vaiheen (4) tarkistus on oleellinen vain, jos $n \leq (\log_2 n)^5$ eli kun $n \leq 5\,690\,033$ - tämän kokoluokan luvut ovat nopeita tarkistaa jo alkeellisimmillakin testeillä. Voimme siis olettaa jatkossa, että algoritmin vaihe (5) palauttaa vastaukseksi ”alkuluku”.

Kiinnitämme nyt luvut n ja r, joista valitsemme luvun r kuten lauseessa 4.9. Olkoon luku p jokin luvun n alkulukujakaja siten, että $\mathrm{ord}_r(p) > 1$. Tällainen alkuluku p on olemassa, sillä jos ei olisi, niin $\mathrm{ord}_r(n) = 1$, mutta näytimme aiemmin, että $\mathrm{ord}_r(n) > (\log_2 n)^2$. Kiinnitämme myös tämän luvun p.

Olkoon jatkossa luku $l := \lfloor \sqrt{\varphi(r)} \log_2 n \rfloor$ vaiheessa (5) käytettävä raja polynomien tarkistusten lukumäärälle. Tutkimme seuraavaksi renkaassa $\mathbb{Z}_\mathrm{p}[X]$ lineaarisia polynomeja $X + a, 0 \leq a \leq l$ ja näiden tulopolynomeja eli joukon

$$P = \left\{ \prod_{a=0}^{l} (X+a)^{\beta_a} \ : \ \beta_a \geq 0 \text{ kaikille } 0 \leq a \leq l \right\} \subset \mathbb{Z}_\mathrm{p}[X]$$

jäseniä. Tarkoituksenamme on osoittaa joukon P polynomien potensseilla olevan erikoisia ominaisuuksia laskettaessa näiden jäännöspolynomi modulo $X^r - 1$.

LEMMA 4.10. *Renkaassa* $\mathbb{Z}_\mathrm{p}[X]$ *kaikille* $0 \leq a \leq l$ *on voimassa* $(X+a)^n \equiv X^n + a \mod (X^r - 1)$.

TODISTUS. Tapaus $a = 0$ on selvä. Olkoon a mikä vain luvuista $\{1, ..., l\}$. Oletuksen (5) nojalla renkaassa $\mathbb{Z}_n[X]$ on voimassa yhtälö $(X+a)^n \equiv X^n+a \mod (X^r-1)$, jolloin on olemassa polynomit $P, Q \in \mathbb{Z}[X]$ siten, että $(X+a)^n - (X^n+a) = (X^r-1)\cdot P + n \cdot Q$. Nyt $p \mid n$, jolloin voimme merkitä polynomia $\hat{Q} := \frac{n}{p}Q$. Tällöin siis $(X+a)^n-(X^n+a) = (X^r-1)\cdot P+p\cdot\hat{Q}$ ja siten $(X+a)^n \equiv X^n+a \mod (X^r-1)$ renkaassa $\mathbb{Z}_p[X]$.

□

LEMMA 4.11. *Renkaassa $\mathbb{Z}_p[X]$ kaikille $0 \leq a \leq l$ ovat voimassa kongruenssiyhtälöt*

$$(X+a)^p \equiv X^p + a \text{ ja } (X+a)^{\frac{n}{p}} \equiv X^{\frac{n}{p}} + a \mod (X^r-1)$$

TODISTUS. Tapaus $a = 0$ on jälleen selvä molempien yhtälöiden kohdalla. Lauseen 4.5 kohdan (b) nojalla $(X+a)^p = X^p + a$ renkaassa $\mathbb{Z}_p[X]$ riippumatta luvusta a, joten polynomit ovat siten kongruentteja modulo $X^r - 1$.

Jälkimmäisen yhtälön todistaaksemme tarvitsemme renkaan $\mathbb{Z}_p[X]$ ideaalia

$$I := \{f \cdot (X^r - 1) \ : \ f \in \mathbb{Z}_p[X]\}.$$

Liikumme jatkossa renkaassa $\mathbb{Z}_p[X]/I$. Oletusten nojalla

$$(X^{\frac{n}{p}} + a)^p = X^n + a = (X+a)^{p\cdot\frac{n}{p}}. \tag{4.3}$$

Olkoot nyt polynomit $f = X^{\frac{n}{p}} + a$ ja $g = (X+a)^{\frac{n}{p}}$, jolloin yhtälön (4.3) nojalla $f^p = g^p$. Tällöin $f^p + (-g)^p = 0$, sillä $(-g)^p = (-1)^p g^p = -g^p$, sillä vaikka olisi $p = 2$, niin tällöin $1 = -1$. Nyt siis lauseen 4.5 (a)-kohdan mukaan $(f-g)^p = f^p+(-g)^p = 0$. Merkitsemme nyt $h = f - g$ ja riittää enää osoittaa, että $h = 0$, eli $h \in I$.

Koska $\text{syt}(p, r) = 1$, niin myöhemmin esiteltävien (ja tästä lauseesta riippumattomien) tulosten 4.17 ja 4.18 nojalla $X^r - 1$ jakaantuu jaottomiin, erillisiin tekijöihin h_i, jolloin

$$X^r - 1 = \prod_i h_i.$$

Koska $h^p \in I$, niin jokainen tekijä h_i jakaa polynomin h^p. Koska tekijät h_i ovat jaottomia, niin $h_i \mid h$ kaikilla indekseillä i. Lemman 4.18 nojalla tekijät h_i ovat keskenään eri polynomeja, joten $X^r - 1 \mid h$ ja väite seuraa.

□

Tarvitsemme jatkossa merkintää $I(u, f)$ lyhenteeksi sille, että seuraavat ehdot ovat voimassa:

(a) $u \geq 1$ ja $f \in \mathbb{Z}_p[X]$
(b) $f^u = f(X^u) \mod (X^r - 1)$ renkaassa $\mathbb{Z}_p[X]$.

Jos ehto $I(u, f)$ pätee, niin sanomme luvun u olevan *introspektiivinen* polynomille f [**1**, s. 4]. Kahden edellisen lemman nojalla pätee $I(u, f)$ ainakin kun $u = n, u = p$ tai $u = \frac{n}{p}$ ja $f = X + a, 0 \leq a \leq l$. Laajennamme seuraavaksi tämän ominaisuuden koskemaan eksponenttien tuloja ja polynomien tuloja.

LEMMA 4.12. *Jos $I(u, f)$ ja $I(v, f)$, niin $I(uv, f)$.*

TODISTUS. Koska $I(v,f)$ on voimassa, pätee $f^v \equiv f(X^v) \mod (X^r-1)$ ja hyödyntäen lemman 4.4 (c)-kohtaa $f^{uv} = (f^v)^u \equiv (f(X^v))^u \mod (X^r-1)$. Koska $I(u,f)$, niin $f^u(X) \equiv f(X^u) \mod (X^r-1)$ eli $f^u - f(X^u) = (X^r-1)g$ jollekin polynomille $g \in \mathbb{Z}_p[X]$. Korvaamalla tässä yhtälössä muuttuja X muuttujalla X^v saamme

$$(f(X^v))^u - f((X^v)^u) = ((X^v)^r - 1)g(X^v),$$

ja koska $(X^r-1)(X^{(v-1)r} + ... + X^r + 1) = X^{rv} - 1$, niin $(X^r-1) \mid (X^{rv}-1)$ ja

$$f^{uv} \equiv (f(X^v))^u \equiv f((X^v)^u) \equiv f(X^{uv}) \mod (X^r-1).$$

Tällöin siis pätee väite $I(uv,f)$.

□

LEMMA 4.13. *Jos $I(u,f)$ ja $I(u,g)$, niin $I(u,fg)$.*

TODISTUS. Oletusten ja lemman 4.4 (b)-kohdan nojalla

$$(fg)^u = f^u \cdot g^u \equiv f(X^u) \cdot g(X^u) = (fg)(X^u) \mod (X^r-1).$$

□

Edellisten lemmojen avulla voimme päätellä ehdon $I(u,f)$ olevan voimassa mielivaltaiselle tulolle lineaarisia polynomeja $(X+a), 0 \le a \le l$ (eli aiemmin määrittelemämme joukon P polynomille f) ja luvulle u, joka on mielivaltainen tulo luvuista $\frac{n}{p}$ ja p. Merkitsemme sopivien eksponenttien joukkoa

$$U = \left\{ \left(\frac{n}{p}\right)^i p^j \ : \ i,j \ge 0 \right\}.$$

Olkoon myös joukko

$$T = \{u \mod r \ : \ u \in U\},$$

joka on joukon $\mathbb{Z}_r^\times$ osajoukko, sillä koska luvut n ja r ovat suhteellisia alkulukuja, niin myös p ja r ovat suhteellisia alkulukuja, mistä seuraa, että kaikki joukon U alkiot ovat kääntyviä kertolaskun modulo r suhteen. Olkoon vielä luku $t := \#T$. Koska $\mathrm{ord}_r(n) > (\log_2 n)^2$ ja $n \in U$, niin $t > (\log_2 n)^2$.

Nyt voimme muotoilla uuden lemman polynomeille $f \in P$ ja eksponenteille $u \in U$:

LEMMA 4.14. *Kaikille polynomeille $f \in P$ ja luvuille $u \in U$ pätee*

$$f^u \equiv f(X^u) \mod (X^r-1)$$

renkaassa $\mathbb{Z}_p[X]$.

TODISTUS. Olkoon f mielivaltainen polynomi joukossa P, eli

$$f = (X+a_1)^{\beta_1} \cdots (X+a_k)^{\beta_k},$$

missä kaikille $i = 1, ..., k$ pätee $1 \le a_i \le l$. Jos $u = 1$, niin mitään todistettavaa ei ole. Olkoon siis $u = (\frac{n}{p})^i p^j$, missä $i,j \ge 0$ ja ainakin toinen luvuista i tai j on aidosti positiivinen. Olkoon nyt $g_i = X + a_i$ ja $f_i = g_i^{\beta_i}, 1 \le i \le k$ sekä $u_{n/p} = (\frac{n}{p})^i$ ja $u_p = p^j$, jolloin $u = u_{n/p} u_p$. Lemmojen 4.10 ja 4.11 nojalla kaikki polynomit $g_i, 1 \le i \le k$ toteuttavat ehdot $I(u_p, g_i)$ ja $I(u_{n/p}, g_i)$, jolloin lemman 4.12 nojalla $I(u, g_i)$.

Soveltamalla β_i kertaa lemmaa 4.13 jokaiselle polynomille g_i saamme, että $I(u, g_i^{\beta_i})$ eli $I(u, f_i)$ jokaiselle $1 \leq i \leq k$. Edelleen saman lemman 4.13 avulla saamme myös polynomille osoitettua $f = f_1 \cdots f_k$ ehdon $I(u, f)$, mikä on yhtäpitävää väitteen kanssa. □

Seuraavaksi tavoitteenamme on löytää polynomin $X^r - 1$ jaoton tekijä h, jonka asteluku on ainakin kaksi. Tällaisen olemassaolo vaikuttaa jo kokeilemalla eri luvuille r melko selvältä, mutta sen osoittaminen vaatii monimutkaisempia algebran rakenteita. Tämän vuoksi sivuutamme muutaman todistuksen. Avuksi jatkoon tarvitsemme seuraavan määritelmän.

MÄÄRITELMÄ 4.15. Sanomme, että ryhmän G alkio g on primitiivinen k. yksikköjuuri, jos $g^k = 1_G$ ja $g^i \neq 1$ kaikille $i = 1, ..., k-1$. Tässä 1_G on ryhmän G neutraalialkio.

MÄÄRITELMÄ 4.16. Olkoon ω primitiivinen r. yksikköjuuri. Sanomme, että

$$Q_r(X) = \prod_{\substack{1 \leq s \leq r \\ \mathrm{syt}(s,r)=1}} (X - \omega^s)$$

on r. syklotominen polynomi.

Selvästi r. syklotominen polynomi Q_r on asteluvultaan $\varphi(r)$. Koska ω on primitiivinen r. yksikköjuuri, niin kaikille luvulle r suhteellisille alkuluvuille $1 \leq s < r$ on ω^s myös primitiivinen r. yksikköjuuri. Syklotominen polynomi on myös tällöin riippumaton primitiivisen yksikköjuuren ω valinnasta riippumatta.

LEMMA 4.17. *$Q_r(X) \mid X^r - 1$ renkaassa $\mathbb{Z}_\mathrm{p}[X]$.*

TODISTUS. Sivuutamme todistuksen, ks. [**13**, ss. 59-62]. □

LEMMA 4.18. *Olkoon $Q_r(X)$ r. syklotominen polynomi renkaassa $\mathbb{Z}_\mathrm{p}[X]$. Tällöin $Q = h_1 \cdots h_s$, missä polynomit $h_1, ..., h_s \in \mathbb{Z}_\mathrm{p}[X]$ ovat jaottomia, suurimpien asteiden termien kertoiminaan 1, keskenään eri polynomeja ja asteluvuiltaan* $\mathrm{ord}_r(p)$.

TODISTUS. Sivuutamme tämänkin todistuksen, ks. [**13**, ss. 59-62]. □

Edellisen lemman nojalla on nyt olemassa $\mathrm{ord}_r(p)$-asteinen jaoton polynomi $h \in \mathbb{Z}_\mathrm{p}[X]$, joka jakaa polynomin $X^{r-1} + ... + X + 1$ ja siten myös polynomin $X^r - 1$. Lemman 4.18 ja luvun p valinnan nojalla polynomin h asteluku on ainakin 2, sillä $\deg(h) = \mathrm{ord}_r(p) > 1$. Kiinnitämme tällaisen polynomin h ja koska h on jaoton, niin saamme muodostettua kunnan $F = \mathbb{Z}_\mathrm{p}[X]/(h)$. Kunnan F kertaluku on nyt p^d, sillä polynomit tässä kunnassa ovat asteluvultaan korkeintaan $d := \deg(h)$ ja polynomien termeillä on kullakin mahdollisia kertoimia p kappaletta.

Muistakaamme aiemmin määrittelemämme lineaaristen polynomien tulopolynomien joukko P. Käytämme edelleen vaiheen (5) tarkistusten lukumäärän ylärajalle merkintää $l = \lfloor \sqrt{\varphi(r)} \log_2 n \rfloor$. Olkoon nyt joukko

$$G := \{f \mod h \ : \ f \in P\} = \{ \prod_{1 \leq a \leq l} (X + a)^{\beta_a} \mod h \ : \ \beta_a \geq 0 \text{ kaikille } 1 \leq a \leq l\}$$

joukon P kuva modulo h. Koska $\deg(h) \geq 2$ ja h on jaoton, niin yksikään polynomeista $f \in P$ ei ole nolla kunnassa F: muuten h olisi joko alle toisen asteen polynomi tai tällaisten tulo. Nyt siis joukon G on oltava kunnan F multiplikatiivisen ryhmän $F^\times = F \setminus \{0\}$ osajoukko. Näytämme seuraavaksi, että polynomit joukosta P ja asteluvultaan alle t kuvautuvat kaikki eri polynomeiksi joukkoon G.

LEMMA 4.19. *Olkoon $f, g \in P$ keskenään eri polynomeja siten, että $deg(f), deg(g) < t$. Tällöin $f \mod h \neq g \mod h$.*

TODISTUS. Koska h on syklotomisen polynomin $Q_r(X)$ tekijä ja $Q_r(X) \mid X^r - 1$, niin $X^r = 1$ kunnassa F ja siten X on r. primitiivinen yksikköjuuri. Todistamme ensimmäisen väitteen todistamalla vastaväitteen vääräksi.

Jos on olemassa eri polynomit $f, g \in P$, joille $\deg(f), \deg(g) < t$ ja $f \equiv g \mod h$, niin $f = g$ kunnassa F. Erityisesti tällöin kaikille luvuille $u \in U$ pätee $f^u = g^u$ kunnassa F. Koska $f, g \in P$ ja $u \in U$, niin $f(X^u) = f^u = g^u = g(X^u)$, jolloin X^u on polynomin $\hat{f} := f - g$ juuri. Koska $T \subset \mathbb{Z}_r^\times$, niin $\text{syt}(u, r) = 1$ kaikille $u \in T$. Lisäksi X on r. primitiivinen yksikköjuuri, jolloin X^u on r. primitiivinen yksikköjuuri kaikille $u \in T$. Erityisesti kaikki nämä potenssit X^u ovat eri juuria jokaiselle $u \in T$, joten polynomilla $\hat{f}$ on ainakin $\#T$ eri juurta. Nyt siis $\deg(\hat{f}) = \#T = t > \deg(\hat{f})$, mikä on ristiriita, sillä F on kunta.

□

LEMMA 4.20. $\#G \geq \binom{t+l}{t-1}$.

TODISTUS. Lemman 4.19 mukaan kaikki polynomit $f \in P$, joiden asteluku on alle t, kuvautuvat eri alkioiksi kuntaan F, eli ne ovat myös eri alkioita joukossa G. Lisäksi yksikään näistä ei ole nolla joukossa G.

Kaksi ensimmäisen asteen polynomia $X + a, X + b \in F$, $a \neq b$, ovat samat silloin, kun $a \equiv b \mod p$, mutta koska $a, b \leq l = \lfloor \sqrt{\varphi(r)} \log_2 n \rfloor \leq \sqrt{r} \log_2 n \underset{(1)}{<} r < p$, niin $X + a$ ja $X + b$ eivät voi olla samat; epäyhtälö (1) seuraa siitä, että $\text{syt}(r, n) = 1$ ja $\text{ord}_r(n) > (\log_2 n)^2$.

Joukon P ensimmäisen asteen polynomeja on siis ainakin $l + 1$ kappaletta. Jos $t = 2$, niin lukemalla alle t -asteisten polynomien joukkoon myös $1 \in P$ saamme ainakin $l + 2 = \binom{l+t}{t-1}$ sopivaa polynomia joukosta P.

Induktiolla voimme laajentaa tämän koskemaan yleistä lukua t: Jos väite pätee alle $t - 1$ -asteisten polynomien lukumäärää laskiessa, niin kohtalaisen helposti voimme huomata, että tasan $(t-1)$-asteisia polynomeja on $\binom{l+t-1}{t-1}$ kappaletta. Tämän voi päätellä siitä, kuinka monella tavalla eksponentit $\beta_a \geq 0, 0 \leq a \leq l$ ovat sijoitettavissa polynomeille $(X + a)$ tulopolynomissa $\prod_{a=0}^{l} (X + a)^{\beta_a}$ siten, että summa $\sum_{a=0}^{l} \beta_a = t - 1$; jokainen β_a on summa ykkösiä, ja riittää laskea, kuinka monella tapaa voimme sijoittaa $t - 1$ ykköstä kaikkiin $l + 1$ kappaleeseen eksponentteja. Induktio-oletuksen

nojalla enintään $(t-2)$-asteisia polynomeja on ainakin $\binom{t-1-l}{t-2}$, joten yhteensä enintään $(t-1)$ -asteisia polynomeja on ainakin $\binom{l+t-1}{t-1} + \binom{t-1-l}{t-2} = \binom{l+t}{t-1}$ kappaletta.

Koska asteluvultaan alle t olevat polynomit $f \in P$ kuvautuvat eri alkioiksi joukkoon G, niin saamme näin joukon G alkioiden lukumääräksi $\#G$ ainakin $\binom{l+t}{t-1}$. □

Seuraava lemma auttaa osoittamaan, että luvun n on oltava luvun p potenssi. Olkoon sitä varten joukko

$$U_0 = \left\{ \left(\frac{n}{p}\right)^i p^j \ : \ 0 \leq i,j \leq \lfloor \sqrt{t} \rfloor \right\}$$

aiemman joukon U osajoukko.

LEMMA 4.21. *Jos luku n ei ole luvun p potenssi, niin $\#G \leq n^{\sqrt{t}}$.*

TODISTUS. Koska n ei ole luvun p potenssi, niin joukon U_0 alkioiden lukumäärän on oltava $\#U_0 = (\lfloor \sqrt{t} \rfloor + 1)^2 > t = \#T$. Voimme nyt valita kaksi joukon U_0 alkiota $u_1 > u_2$ siten, että ne ovat samat modulo r, jolloin $u_1 = u_2 + kr$ jollekin luvulle k. Tällöin

$$X^{u_1} - X^{u_2} = X^{u_2}(X^{kr} - 1) = X^{u_2}(X^{(k-1)r} + ... + X^r + 1)(X^r - 1) \equiv 0 \mod (X^r - 1),$$

jolloin $X_1^u \equiv X^{u_2} \mod (X^r - 1)$. Jos valitsemme mielivaltaisen polynomin $f \in P$, niin koska $u_1, u_2 \in U$, pätee $f^{u_1} \equiv f(X^{u_1}) \equiv f(X^{u_2}) \equiv f^{u_2} \mod (X^r - 1)$. Koska $h \mid (X^r - 1)$, niin kunnassa F polynomilla $X^{u_1} - X^{u_2}$ on ainakin $\#G$ juurta. Tällöin siis $\#G \leq u_1 \leq \max U_0 = (\frac{n}{p})^{\lfloor \sqrt{t} \rfloor} p^{\lfloor \sqrt{t} \rfloor} = n^{\lfloor \sqrt{t} \rfloor}$. □

LEMMA 4.22. *Luvuille $m, n \in \mathbb{N}$ pätee:*

(i) $\binom{m+1}{n+1} > \binom{m}{n}$
(ii) $\binom{m+1}{n} > \binom{m}{n}$
(iii) $\binom{2k+1}{k} > 2^{k+1}$ *kaikille* $k \in \mathbb{N} \setminus \{1\}$.

TODISTUS. Kohdat (i) ja (ii) ovat yleistietoa, joten käsittelemme vain kohdan (iii). Selvästi

$$\binom{2k+1}{k} = \frac{(2k+1)(2k)\cdots(k+2)(k+1)}{k(k-1)\cdots 2 \cdot 1},$$

missä $2k+1 > 2k, 2k > 2(k-1), \ldots, k+3 > 4, k+2 > 2$ ja $k+1 > 2$. Niinpä

$$\begin{aligned} \tfrac{(2k+1)(2k)\cdots(k+2)(k+1)}{k(k-1)\cdots 2\cdot 1} &= \tfrac{(2k+1)(2k)\cdots(k+2)}{k(k-1)\cdots 2\cdot 1} \cdot (k+1) \\ &> \tfrac{(2k)(2(k-1))\cdots 4\cdot 2}{k(k-1)\cdots 2\cdot 1} \cdot 2 \\ &= 2^{k+1}. \end{aligned}$$

□

Nyt voimme todistaa testin tuloksen oikeellisuuden loppuun:

LAUSEEN 4.7 TODISTUKSEN LOPPU. Olettaen, että AKS-testi toteaa luvun n olevan alkuluku, niin lemmojen 4.19 ja 4.22(i) mukaan luvuille $t = \#T$ ja $l = \lfloor \sqrt{\varphi(r)} \log_2 n \rfloor$ on voimassa

$$\#G \geq \binom{t+l}{t-1} \geq \binom{l+1+\lfloor \sqrt{t} \log_2 n \rfloor}{\lfloor \sqrt{t} \log_2 n \rfloor},$$

sillä $t > (\log_2 n)^2$ ja siten $t > \sqrt{t} \log_2 n \geq \lfloor \sqrt{t} \log_2 n \rfloor$. Edelleen lemman 4.22(ii) avulla voimme päätellä, että

$$\binom{l+1+\lfloor \sqrt{t} \log_2 n \rfloor}{\lfloor \sqrt{t} \log_2 n \rfloor} \geq \binom{2\lfloor \sqrt{t} \log_2 n \rfloor + 1}{\lfloor \sqrt{t} \log_2 n \rfloor}$$

sillä koska $T \subset \mathbb{Z}_r^\times$, niin $t \leq \varphi(r)$ ja tällöin myös $l = \lfloor \sqrt{\varphi(r)} \log_2 n \rfloor \geq \lfloor \sqrt{t} \log_2 n \rfloor$. Koska $\sqrt{t} > \log_2 n$, niin $\lfloor \sqrt{t} \log_2 n \rfloor > \lfloor (\log_2 n)^2 \rfloor \geq 1$, jolloin lemman 4.22(iii) nojalla

$$\binom{2\lfloor \sqrt{t} \log_2 n \rfloor + 1}{\lfloor \sqrt{t} \log_2 n \rfloor} > 2^{\lfloor \sqrt{t} \log_2 n \rfloor + 1} \geq 2^{\sqrt{t} \log_2 n} = n^{\sqrt{t}}.$$

Lemman 4.21 mukaan $\#G \leq n^{\sqrt{t}}$, jos luku n ei ole luvun p potenssi, joten nyt luvun n on oltava luvun p potenssi. Vaiheen (1) läpäisyn nojalla $n = p$, joten luku n todellakin on alkuluku. □

4.4. AKS-testin aikavaativuus

Olemme edellisessä kappaleessa osoittaneet AKS-alkulukutesti deterministiseksi, eli se antaa täysin varman vastauksen. Seuraavaksi on vielä osoitettava, että se toimii polynomisessa ajassa.

Tässä aika-arviossa hyödynnämme myös alussa käsittelemäämme $\mathcal{O}^\sim$-merkintää siistiäksemme kertaluokkaan vain vähän vaikuttavia sisäkkäisiä logaritmeja pois. Jos implementoimme mukaan myös tehokkaammat kerto- ja jakolaskualgoritmit, niin saamme pudotettua näiden laskutoimitusten kertaluokkaa: kahden luvun, jotka ovat enintään n, kerto- ja jakolaskut ovat tällöin $\mathcal{O}^\sim(\log n)$-luokan toimenpiteitä.

AKS-algoritmissa on monta pitkää vaihetta, joten jaamme tehtävän seuraavasti osiin. Yhdistämme hieman vaiheita (2)-(4), sillä luvun n kertaluvun modulo r laskeminen ei tässä toteutuksessa onnistu, jos luvut n ja r eivät ole suhteellisia alkulukuja. Tämä olisi toki mahdollista tehdä myös poikkeuskäsittelijällä.

```
// Testaa AKS-alkulukutestillä, onko annettu luku alkuluku
static bool onAlkuluku_AKS(BigInteger n)
{
  // Tarkistamme, onko n jonkin kokonaisluvun potenssi.
  if (onKokonaislukupotenssi(n)) return false;

  // Etsimme sopivan luvun r, jolle ord_r (n) > log^2 n.
  // Teemme syt-testin samalla.
  // Palauttaa -1, jos 1<syt(r,n)<n jollekin ehdokasluvulle r.
  // Palauttaa r = n, jos sopivaa kertalukua ei löydy ja n
     pieni.
```

```
  BigInteger r = EtsiR(n);

  // Syt-tarkistus.
  if (r == -1) return false;

  // Jos n on pieni, niin syt-testi riittää toteamaan luvun n
     alkuluvuksi.
  if (R >= N) return true;

  // Polynomitesti.
  return PolynomitestinTulos(N, R);
}
```

4.4.1. Vaihe 1: kokonaislukupotenssi. Ensimmäisenä on muodostettava algoritmi sen selvittämiselle, onko luku n alkulukupotenssi. Seuraava C#-kielelle käännetty algoritmi on peräisin lähdemateriaalista ja eräs tapa saavuttaa tavoitteemme [**5**, s. 21]

```
// Selvittää, onko syöteluku n jonkin kokonaisluvun potenssi
static bool onKokonaislukupotenssi(BigInteger n)
{
    BigInteger a, c, m;
    int b;
    b = 2;
    while (b <= BigInteger.Log(n) / Math.Log(2) )
    {
        a = 1;
        c = n;
        while (c - a >= 2)
        {
            m = (a + c) / 2;
            BigInteger p = BigInteger.Min(BigInteger.Pow(m,b), n
               + 1);
            if (p == n) return true;
            if (p < n) a = m;
            else c = m;
        }
        b += 1;
    }
    return false;
}
```

Algoritmin ideana on suorittaa binäärihaku testaten eri kantalukuja eksponenteilla $b = 2, 3, 4, ..., \lfloor \log_2 n \rfloor$. Mahdollisia eksponentteja on siis lukuun n nähden logaritminen määrä, ja ideana on myös tehdä kantalukujen haku logaritmisesti iteroiden lähemmäksi toimivaa ratkaisua, eikä yksi kerrallaan kokeilemalla.

Algoritmissa käymme ulomman while-silmukan läpi enintään $\mathcal{O}(\log n)$ kertaa, jonka sisällä olevan while-silmukan operaatiot ovat oleellisimpia kokonaissuoritusajan kannalta, sillä muuttujien alustaminen on vakioaikaista. Koska luku b ei koskaan kasva suuremmaksi kuin $\log_2 n$, niin ykkösen lisääminen ulomman silmukan lopussa on enintään $\mathcal{O}(\log b) = \mathcal{O}(\log \log n)$ -luokan operaatio. Sisemmässä silmukassa jokaisella kierrokselle luku m ainakin puolittuu lähtien aloitusluvusta $\frac{n+1}{2}$, joten tässäkin suoritamme enintään logaritmisen määrän laskutoimituksia.

Sisemmän silmukan ensimmäinen laskutoimitus on enintään $\mathcal{O}(\log^2 n)$-luokkaa. Luku $m^b \leq \frac{(n+1)^2}{4} = \mathcal{O}(n)$, jolloin minimin luvuista m^b ja $n+1$ selvittäminen vaatii toistettua neliöintiä käyttäen enintään $\mathcal{O}(\log^3 \mathcal{O}(n)) = \mathcal{O}(\log^3 n)$ laskutoimitusta, sillä m^b ei kasva kovin isoksi. Tämän jälkeen edessä on lähinnä vakioaikaisia toimenpiteitä.

Kokonaisuudessaan AKS-algoritmin ensimmäinen on siis kertaluokaltaan enintään

$$\mathcal{O}(\log n) \cdot (\mathcal{O}(\log n) \cdot (\mathcal{O}(\log^2 n) + \mathcal{O}(\log^3 n)) + \mathcal{O}(\log \log n)) = \mathcal{O}(\log^5 n).$$

Tehokkaammalla kertolaskualgoritmilla ja huolellisemmalla analysoinnilla tämä raja on alennettavissa jopa $\mathcal{O}^{\sim}(\log^3 n)$-luokkaan [**1**, s. 6][**5**, s. 21], mutta kokonaissuoritusajassa tämä ei ole vielä merkityksellisin osa algoritmia.

4.4.2. Vaiheet 2-4: Sopivan luvun r löytäminen. Sopivan kokoisen luvun r löytäminen on tärkein osa algoritmin lopullista suoritusaikaa muodostettaessa: itse luvun löytäminen käy suhteellisen nopeasti, mutta jos r on liian iso, niin vaiheessa (5) AKS-ehto (4.2) on arvioitava hyvin suurelle määrälle polynomeja $X + a, 1 \leq a \leq \lfloor \varphi(r) \log_2 n \rfloor$. Esitämme algoritmin vaiheet (2)-(4) yhdistettyinä seuraavasti:

```
// Etsii luvun r siten, että ord_r (n) > (log_2 n)^2
static BigInteger EtsiR(BigInteger n)
{
    double x = BigInteger.Log(n, 2);
    BigInteger alaraja = BigInteger.Parse(
      Math.Floor(x * x).ToString()
    );
    BigInteger ylaraja = BigInteger.Parse(
      (Math.Ceiling(Math.Pow(x, 5))).ToString()
    );
    if (ylaraja > n) ylaraja = n;
    BigInteger r;
    for (r = 2; r <= ylaraja; r++)
    {
      BigInteger syt = syt(n, r);
      if (syt > 1 && syt < n) return -1; // Jos luku n ei
         viritä aliryhmää modulo r
      if (KertalukuTarpeeksiSuuri(n, r, alaraja)) return r;
```

```
    }
    return r; // Tapahtuu, jos n<=(log_2 n)^5
}

// Selvittää, onko ord_r (n)>alaraja, jos syt(r,n)=1
// Tapauksessamme alaraja = (log_2 n)^2.
static BigInteger KertalukuTarpeeksiSuuri(BigInteger n,
    BigInteger r, BigInteger alaraja)
{
  BigInteger k = 1;
  BigInteger x = n % r;
  BigInteger a = x;
  while (a != 1)
  {
    a = a * x % r;
    k++;
    if (a == 0) return false;
  }
  return (k > alaraja);
}
```

Totesimme aiemmin, että luku r on korkeintaan $\lceil (\log_2 n)^5 \rceil = \mathcal{O}(\log^5 n)$. Tämän myötä siis luvun r löytäminen on toteutettavissa silmukalla, jossa kokeilemme luvulle r vaihtoehtoja $2, 3, ..., \lceil (\log_2 n)^5 \rceil$, mikä vaatii noin $\mathcal{O}(\log^5 n)$ kierrosta. Ensin vaiheen (3) mukaisesti tarkistamme ehdokkaan r ja luvun n suurimman yhteisen tekijän: jos $1 < \text{syt}(r, n) < n$, niin kertaluvun laskeminen on hyödytöntä, sillä n ei ole tällöin alkuluku. Jos suurimman yhteisen tekijän tarkistus on kohdallaan, niin tutkimme, millä luvun i arvolla pätee $n^i \equiv 1 \mod r$. Kun jollekin eksponentille $i > (\log_2 n)^2$ eikä yhtälö edelleenkään päde, niin tiedämme että $\text{ord}_r(n) > (\log_2 n)^2$. Jokainen silmukan kierros siis vaatii suurimman yhteisen tekijän tarkistuksen osalta $\mathcal{O}(\log^3 n)$ pientä laskutoimitusta ja kertaluvun modulo r suuruusluokan selvittämisen osalta $\mathcal{O}(\log^2 n)$ kappaletta kerto- ja jakolaskuja luvuilla, joiden kokoluokka on enintään $\mathcal{O}(\log^5 n)$. Yhteensä kertaluvun laskeminen siis on kertaluokaltaan

$$\begin{aligned} \mathcal{O}(\log^2 n)(2 \cdot \mathcal{O}(\log^2 \log^5 n)) &= \mathcal{O}(\log^2 n) \cdot \mathcal{O}((5 \cdot \log\log n)^2) \\ &= \mathcal{O}((\log^2 n)(\log\log n)^2) \\ &= \mathcal{O}^{\sim}(\log^2 n). \end{aligned}$$

Lisätessämme tähän suurimman yhteisen tekijän laskemisen $\mathcal{O}(\log n)$-määrällä $\mathcal{O}(\log n \log r) = \mathcal{O}(\log n \log\log^5 n)$-laskutoimituksia saamme yhteisajaksi silmukan jokaiselle kierrokselle

$$\mathcal{O}(\log n) \cdot \mathcal{O}(\log n \log\log^5 n) + \mathcal{O}^{\sim}(\log^2 n) = \mathcal{O}(\log^2 n).$$

Tehokkaampien kerto- ja jakolaskualgoritmien lisääminen tähän ei oleellisesti muuta suoritusaikaa, sillä tällöin ainoastaan sisäkkäiset logaritmit lisääntyvät.

Koska suoritamme ulomman silmukan $\mathcal{O}(\log^5 n)$ kertaa, niin saamme vaiheiden (2)-(4) yhteiskestoksi

$$\mathcal{O}(\log^5 n) \cdot \mathcal{O}^{\sim}(\log^2 n) = \mathcal{O}^{\sim}(\log^7 n).$$

4.4.3. Vaihe 5: polynomilaskenta. Viides ja viimeinen vaihe on laskennallisesti kaikkein vaativin. Tehtävänämme on arvioida ehdon

$$(X + a)^n \mod (X^r - 1) = X^{n \mod r} + a$$

pitävyyttä luvuille $a = 1, 2, ..., \lfloor \sqrt{\varphi(r)} \log_2 n \rfloor$. Algoritmin kannalta on hyvä tallentaa polynomit järjestettynä taulukkona suurista kokonaisluvuista, matemaattisesti siis vektoreina eri X^i-termien kertoimista.

Muotoilemme polynomitestin ohjelmakoodina seuraavasti:

```
// Tarkistaa, onko (X+a)^n mod (X^r-1) = X^(n mod r) + a
   kaikille 1<=a<=floor(sqrt(phi(r))*log_2 n)
static bool PolynomitestinTulos(BigInteger n, BigInteger R)
{
  // (Ei toimi, jos r ei riittävän pieni)
  int r = int.Parse(R.ToString());
  // Laskemme Eulerin phi-funktion arvon luvulle r
  int phi_r = int.Parse(EulerPhi(R).ToString());

  // Määritämme ylärajan luvulle a
  int ylaraja =
     (int)(Math.Floor(Math.Sqrt(phi_r)*BigInteger.Log(N,2)));

  int a = 1;

  // Teemme polynomitestin kaikille 1 <= a <=
  for (a = 1; a <= ylaraja; a++ )
  {
    // Laskemme (X+a)^n mod (X^r-1)
    BigInteger[] jaannospolynomi =
       ToistettuNeliointiPolynomeille(new BigInteger[] { a, 1
       }, n, n, r);

    int nModr = int.Parse((n % r).ToString());

    // Tarkistamme, että kertoimet ovat mitä halusimme
    if (jaannospolynomi[nModr] != 1 || jaannospolynomi[0] !=
       a % n) return false;
    for (int i = 0; i < jaannospolynomi.Length; i++)
    {
      if (jaannospolynomi[i] != 0 && i != 0 && i != nModr)
         return false;
    }
```

```
  }
  // Jos selvisimme tähän asti, niin luku n on alkuluku
  return true;
}

// Laskee Eulerin phi-funktion arvon luvulle r
static BigInteger EulerPhi(BigInteger r)
{
  BigInteger phi = r - 1;
  BigInteger a = r - 1;
  while (a > 1)
  {
    if (syt(a, r) > 1) phi--;
    a--;
  }
  return phi;
}
```

HUOMAUTUS 4.23. Testin toteutuksessamme luvun r on oltava ”pieni” kokonaisluku, eli enintään $2^{31} - 1 = 2\ 147\ 483\ 647$; jos r on tätä suurempi, niin polynomin tallentaminen ei enää onnistu taulukkona BigInteger-luokan kokonaislukuja kaikkien taulukoiden enimmäispituuden ollessa $2^{31}-1$. Käytännössä tästä ei tule ongelmaa, sillä usein luku r on huomattavasti pienempi kuin $\lceil(\log_2 n)^5\rceil$. Toinen vaihtoehto polynomin tallentamiseksi on myös linkitetty lista, mutta tällaisen tietorakenteen ohjelmointi olisi hieman hankalampaa. Sallittaessa suuret luvut r testin vaatima muisti kasvaa myös erittäin suureksi, sillä ilman älykkäitä tallennusalgoritmeja jo pienillä kertoimilla r-asteisen polynomin tallennus vaatii ainakin r tavua muistia. Jos $r > 2^{31} - 1$, niin tämä tarkoittaa noin kahta gigatavua.

Polynomitestissämme on suoritusajan vähentämiseksi suurille luvuille n hyvä laskea luku $\varphi(r)$. Voisimme toki käyttää myös arviota $\varphi(r) \leq r - 1$, mutta koska polynomien kertolasku osoittautuu erittäin työlääksi, niin yritämme vähentää tällä pienellä toimenpiteellä lopullisten laskutoimitusten määrää suurelle luvulle n. Eulerin φ-funktion arvon luvulle r saamme helpoiten selville laskemalla suurimpia yhteisiä tekijöitä $\mathrm{syt}(a, r)$, missä $2 \leq a \leq r - 1$. Suurimpien yhteisten tekijöiden $\mathrm{syt}(a, r)$ tarkistus enintään kokoluokan $r = \mathcal{O}(\log^5 n)$ luvuille $2 \leq a \leq r - 1$ on kertaluokaltaan enintään

$$(r-2) \cdot \mathcal{O}(\log^3 r) = \mathcal{O}(r \log^3 r) = \mathcal{O}(\log^5 n \log^3 \log^5 n) = \mathcal{O}^{\sim}(\log^5 n).$$

Voimme hyödyntää nyt jäännöspolynomin $(X + a)^n \mod (X^r - 1)$ laskemiseksi toistettua neliöintiä, ja jotta laskettavat polynomit pysyisivät kohtuullisen kokoisina, niin vähennämme niitä modulo $X^r - 1$. Tällöin polynomien asteluku on aina enintään $r - 1$. Polynomien kertolasku on edelleen kuitenkin erittäin hidasta ja kahden n asteisen $\mathbb{Z}_n$-polynomin kertolasku vaatii normaalisti noin $\mathcal{O}(n^2)$ kappaletta yhteen-, kerto- ja jakolaskutoimituksia luvun n kokoisilla luvuilla (katso huomautus 4.24 tehokkaammasta algoritmista). Koska luku r on logaritminen lukuun n nähden, on tämä kuitenkin tapauksessamme $\mathcal{O}((\log^5 n)^2) = \mathcal{O}(\log^{10} n)$. Yhteensä saamme siis polynomien

kertolaskun kertaluokaksi

$$\mathcal{O}(\log^{10} n) \cdot (\mathcal{O}(\log n) + 2 \cdot \mathcal{O}(\log^2 n)) = \mathcal{O}(\log^{12} n).$$

Paremmilla kerto- ja jakolaskualgoritmeilla tämä putoaa luokkaan $\mathcal{O}^{\sim}(\log^{11} n)$.

```
// Laskee kahden Z_n-polynomin P ja Q tulon modulo (X^r-1)
static BigInteger[] PolynomienTuloMod(BigInteger[] P,
   BigInteger[] Q, BigInteger n, int r)
{
  BigInteger[] tulopolynomi = new BigInteger[P.Length +
     Q.Length - 1];
  for (int i = 0; i < tulopolynomi.Length; i++ )
     tulopolynomi[i] = 0;

  for (int i = 0; i < P.Length; i++)
    for (int j=0; j < Q.Length; j++)
      tulopolynomi[i+j] = (tulopolynomi[i+j] + P[i] * Q[j]) %
         n;

  if (r < tulopolynomi.Length)
    tulopolynomi = LaskeJaannosPolynomi(tulopolynomi, n, r);

  return tulopolynomi;
}
```

Jotta voisimme laskea riittävän pieniasteisilla polynomeilla, on myös jokaisen polynomilaskun lopussa tehtävä tarvittaessa vähennys modulo $X^r - 1$. Tämä on kuitenkin helppoa, sillä koska $X^r \equiv 1 \mod (X^r - 1)$, niin voimme korvata tulopolynomin kaikki X^i-termien loppuosan uudella termillä $X^{i \bmod r}$ ja laskea kertoimet samoilla loppuosilla yhteen, kuten tavallisesti polynomien yhteenlaskussa.

```
// Laskee annetun Z_n-polynomin jäännöspolynomin modulo
   (X^r-1)
static BigInteger[] LaskeJaannosPolynomi(BigInteger[]
   polynomi, BigInteger n, int r)
{
  BigInteger[] jaannospolynomi = new BigInteger[r];
  for (int i = 0; i < r; i++ )
  {
    jaannospolynomi[i] = 0;
    int j = i;
    while (j < polynomi.Length)
    {
      jaannospolynomi[i] = (jaannospolynomi[i] + polynomi[j])
         % n;
      j += r;
    }
  }
```

```
  return jaannospolynomi;
}
```

Koska käsittelemämme polynomit ovat jokaisessa vaiheessa näin enintään $r-1$-asteisia, ovat modulo (X^r-1) -vähennettävät polynomit aina enintään $2(r-1)$-asteisia. Tällöin riittää, että laskemme vain $2r$-kappaletta kertoimia yhteen ja vähennämme näitä modulo n. Tapauksessamme siis tämän vaiheen kertaluokka on

$$2r \cdot (\mathcal{O}(\log n) + \mathcal{O}(\log^2 n)) = \mathcal{O}(r \log^2 n) = \mathcal{O}((\log^5 n)(\log^2 n)) = \mathcal{O}(\log^7 n).$$

ja tehokkammalla jakolaskulla

$$2r \cdot (\mathcal{O}(\log n) + \mathcal{O}^{\sim}(\log n)) = \mathcal{O}^{\sim}(r \log n) = \mathcal{O}^{\sim}(\log^6 n).$$

Nyt voimme yhdistää polynomien kertolaskualgoritmimme polynomin toistetun neliöinnin toteutukseemme.

```
// Laskee Z_m-jäännöspolynomin P^n mod (X^r-1)
static BigInteger[]
   ToistettuNeliointiPolynomeille(BigInteger[] P, BigInteger
   n, BigInteger m, int r)
{
  if (n == 1) return P;
  if (n.IsEven)
  {
    BigInteger[] Q = PolynomienTulo(P, P, m, r);
    return ToistettuNeliointiPolynomeille(Q, n/2, m, r);
  }
  else
  {
    BigInteger[] Q = PolynomienTulo(P, P, m, r);
    return PolynomienTulo(P,
       ToistettuNeliointiPolynomeille(Q, (n-1)/2, m, r), m, r);
  }
}
```

Kuten tavallisten kokonaislukujen toistettu neliöinti, niin myös polynomien toistettu neliöinti vaatii enintään $\mathcal{O}(\log n)$ vaihetta. Kokonaisuudessaan jäännöspolynomin $(X+a)^n \mod (X^r-1)$ laskeminen on siis kertaluokaltaan

$$\mathcal{O}(\log n) \cdot (\mathcal{O}(\log^7 n) + \mathcal{O}(\log^{12} n)) = \mathcal{O}(\log^{13} n)$$

ja tehostettuna

$$\mathcal{O}(\log n) \cdot (\mathcal{O}^{\sim}(\log^6 n) + \mathcal{O}^{\sim}(\log^{11} n)) = \mathcal{O}^{\sim}(\log^{12} n).$$

HUOMAUTUS 4.24. Käytimme edellä $\mathcal{O}(r^2)$ laskutoimitusta vaativaa r-asteisten polynomien kertolaskualgoritmia, mikä lisää algoritmin kokonaiskertaluokkaa runsaasti, vaikka käytämmekin vain $r = \lceil (\log_2 n)^5 \rceil$-asteisia polynomeja. On olemassa myös vaikeampi nopeaan Fourier-muunnokseen perustuva polynomien kertolaskualgoritmi, joka tekee saman työn noin $\mathcal{O}(r \log n)$-määrällä laskutoimituksia [**1**, s. 6] [**8**,

s. 233]. Tämän sijoittaminen naiivin polynomien kertolaskualgoritmin sijaan pudottaa jäännöspolynomin laskemisen kertaluokan

$$\mathcal{O}(r \log n) \cdot \mathcal{O}^{\sim}(\log n) = \mathcal{O}^{\sim}(\log^7 n)$$

ongelmaksi.

Suoritamme jäännöspolynomin laskemisen

$$\lfloor \sqrt{\varphi(r)} \log_2 n \rfloor = \mathcal{O}(\sqrt{r} \log n) = \mathcal{O}(\log^{7/2} n)$$

kappaleelle eri polynomeja $(X + a)$. Jäännöspolynomin laskemisen jälkeen tarksitamme myös noin r kappaletta kertoimia, mikä on selvästi kertaluokaltaan noin $\mathcal{O}(r) = \mathcal{O}(\log^5 n)$. Saamme polynomitestin kokonaisajaksi siis

$$\mathcal{O}^{\sim}(\log^5 n) + \mathcal{O}(\log^{7/2} n) \cdot (\mathcal{O}(\log^{13} n) + \mathcal{O}(\log^5 n)) = \mathcal{O}(\log^{33/2} n) = \mathcal{O}(\log^{16.5} n).$$

Huomatuksen 4.24 mukaisella polynomien kertolaskun nopeutuksella suoritusaika on suunnilleen

$$\mathcal{O}^{\sim}(\log^5 n) + \mathcal{O}(\log^{7/2} n) \cdot (\mathcal{O}^{\sim}(\log^7 n) + \mathcal{O}(\log^5 n)) = \mathcal{O}^{\sim}(\log^{21/2} n) = \mathcal{O}^{\sim}(\log^{10.5} n).$$

Selvästi polynomitestin aika-arvio ylittää kaikkien muiden AKS-testin osien kertaluokat, joten nämä ovat myös testin kokonaissuoritusajat. AKS-testi siis on paitsi deterministinen, myös polynomisessa ajassa toimiva alkulukutesti.

4.4.4. Parannukset suoritusaikaan. Ilmiselvin tapa arvioida testin suoritusaikaa alaspäin olisi alentaa maksimia luvulle r, sillä $\lceil \log^5 n \rceil$ on edelleen tarpeettoman suuri. Käytännössä suurillekin luvuille n luku r vaikuttaa olevan pieni, esimerkiksi luvulle $n = 546\ 645\ 644\ 341\ 345\ 484\ 615\ 187\ 311\ 313\ 311\ 313\ 171\ 113$ löytyy $r = 19237$, jolloin $r \approx \log^{1.99911701239} n$. Myös AKS-artikkelin kirjoittajat toteavat, että tieto $r = \mathcal{O}(\log^2 n)$ olisi paras mahdollinen parannus ja auttaisi laskemaan algoritmin kertaluokkaan $\mathcal{O}^{\sim}(\log^6 n)$.

Hendrik W. Lenstra ja Carl Pomerance ovat luoneet AKS-testin pohjalta oman alkulukutestinsä, jonka suoritusaika on noin $\mathcal{O}^{\sim}(\log^6 n)$ [**12**].

LUKU 5

Yhteenveto

Saimme edellisissä kappaleissa aikaiseksi useita testejä hyvin vaihtelevilla suoritusajoilla. Tiivistämme tulokset taulukkoon 1. Taulukossa n on testattava luku ja probabilististen testien kohdalla luku k ilmoittaa testattujen kantalukujen lukumäärän.

Algoritmi	**Tyyppi**	**Suoritusaika**	**Muuta**
Jakolasku	Deterministinen	$\mathcal{O}(\sqrt{n}\log^2 n)$	
Wilsonin lause	Deterministinen	$\mathcal{O}(n\log^2 n)$	
Fermat'n lause	Probabilistinen	$\mathcal{O}(k\log^3 n)$	Epäonnistuu Carmichaelin lukujen kohdalla.
Miller & Rabin	Probabilistinen	$\mathcal{O}(k\log^3 n)$	Jos testattava luku on yhdistetty, niin virhe tapahtuu todennäköisyydellä $1/4^k$.
Solovay & Strassen	Probabilistinen	$\mathcal{O}(k\log^3 n)$	Jos testattava luku on yhdistetty, niin virhe tapahtuu todennäköisyydellä $1/2^k$.
Proth	Deterministinen	$\mathcal{O}(n\log^3 n)$	Toimii vain Prothin luvuille $n = c \cdot 2^m + 1$. Keskimääräinen suoritusaika huomattavasti alhaisempi ja lähellä Solovayn ja Strassenin testiä.
Lucas & Lehmer	Deterministinen	$\mathcal{O}(\log^3 n)$	Toimii vain luvuille $n = 2^p - 1$, missä p on alkuluku.
AKS	Deterministinen	$\mathcal{O}^{\sim}(\log^{10.5} n)$	Vaatii toimiakseen kuvatussa ajassa tehokkaan polynomien kertolasku-algoritmin.

Taulukko 1: Taulukko alkulukutesteistä

LIITE A

Merkintöjä

Merkintä	*Selitys*
$\mathbb{N}$	Luonnollisten lukujen joukko $\{1, 2, 3, \dots\}$.
	Huomaa, että nolla ei ole luonnollinen luku tässä tutkielmassa.
$\mathbb{Z}$	Kokonaislukujen joukko $\{..., -2, -1, 0, 1, 2, 3, ...\}$.
$\mathbb{Q}$	Rationaalilukujen joukko.
$\mathbb{R}$	Reaalilukujen joukko.
$\mathbb{C}$	Kompleksilukujen joukko.
$\mathbb{P}$	Alkulukujen joukko $\{2, 3, 5, 7, ...\}$.
$\mathbb{Z}_n$	Luvun n määräämä jäännösluokkarengas, eli joukko $\{[0]_n, [1]_n, \dots, [n-1]_n\}$.
	Käytetään myös merkintää $\mathbb{Z}/n\mathbb{Z}$.
$\log n$	Luonnollinen (e-kantainen) logaritmi luvulle n.
$\lfloor x \rfloor$	Ns. lattiafunktion (engl. *floor function*) arvo reaaliluvulle x,
	eli lukua x suurin pienempi tai yhtä suuri kokonaisluku $\max\{k \in \mathbb{Z} : k \leq x\}$.
$\lceil x \rceil$	Ns. kattofunktion (engl. *ceiling function*) arvo reaaliluvulle x,
	eli lukua x pienin suurempi tai yhtä suuri kokonaisluku $\min\{k \in \mathbb{Z} : k \geq x\}$.
$\mathrm{ord}(a)$	Luvun a kertaluku, eli jonkin ryhmän alkion a virittämän aliryhmän koko.
$\mathrm{ord}_n(a)$	Luvun a kertaluku modulo n, eli luvun a virittämän aliryhmän koko
	ryhmän $\mathbb{Z}_\mathrm{n}$ alkiona.

Kirjallisuutta

[1] Manindra Agrawal, Neeraj Kayal ja Nitin Saxena: *PRIMES is in P*. Department of Computer Science & Engineering, Indian Institute of Technology Kanpur, Kanpur-208016, India [WWW]. [Viitattu 1.11.2015] Saatavissa: http://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf.

[2] Mikhail J. Atallah ja Marina Blanton: *Algorithms and Theory of Computation Handbook: Special Topics And Techniques*. toinen laitos, CRC Press, 2009

[3] J. W. Bruce: *A Really Trivial Proof of the Lucas-Lehmer Test*. The American Mathematical Monthly, Vol. 100, No. 4 (Apr., 1993), 370-371, 1993.

[4] Johannes A. Buchmann: *Introduction To Cryptography*. Toinen, korjattu laitos (2002), Springer, 2001.

[5] Martin Dietzfelbinger: *Primality Testing In Polynomial Time*. Springer-Verlag Berlin Heidelberg, 2004.

[6] Graham Everest ja Thomas Ward: *An Introduction to Number Theory*. Springer-Verlag London Limited, 2005.

[7] Richard Fitzpatrick: *Euclid's Elements of Geometry*. Käännös J.L.Heibergin kreikankielisestä tekstistä (1883-1885), korjattu versio vuodelta 2008 [WWW]. (Viitattu 20.11.2015) Saatavissa: http://farside.ph.utexas.edu/Books/Euclid/Elements.pdf.

[8] J. von zur Gathen ja J. Gerhard: *Modern Computer Algebra*. Toinen laitos. Cambridge University Press, 2003

[9] G. H. Hardy ja E. M. Wright: *An Introduction To The Theory Of Numbers*. Viides laitos, Oxford University Press, 1979.

[10] C. Y. Hsiung: *Elementary Theory of Numbers*. World Scientific, 1992.

[11] Lassi Kurittu: *Lukuteoria 2*. Jyväskylän yliopisto [WWW]. [Viitattu 25.9.2015] Saatavissa: http://users.jyu.fi/∼lkurittu/lukuteoria2.pdf

[12] H. W. Lenstra jr. ja C. Pomerance: *Primality testing with Gaussian periods*. [WWW] [Viitattu 23.11.2015] Saatavissa: https://math.dartmouth.edu/∼carlp/aks041411.pdf

[13] Rudolf Lidl ja Harald Niederreiter: *Introduction to finite fields and their applications*. Cambridge University Press, 1986. Saatavissa: http://math.boisestate.edu/∼liljanab/MATH508/FiniteFields_and_Applications.pdf

[14] M. Nair: *On Chebyshev-Type Inequalities for Primes*. The American Mathematical Monthly Vol. 89, No. 2, s. 126-129, helmikuu 1982

[15] Oystein Ore: *Number Theory and Its History*. Dover Edition (1988), Courier Corporation, 1948

[16] Jouni Parkkonen: *Algebra*. Jyväkylän yliopisto, 2014 [WWW]. [Viitattu 24.11.2015] Saatavissa: http://users.jyu.fi/∼parkkone/Algebra2014/Algebra2014.pdf

[17] Michael O. Rabin: *Probabilistic algorithm for testing primality*. Journal Of Number Theory 12, 128-138, 1980.