

## 10. Satunnaisalgoritmit

- Probabilistic algorithms, randomized algorithms
- Toinen tapa liiallisen laskennallisen vaativuuden kanssa toimeen tulemiseksi ovat **satunnaisalgoritmit**
- Jotkin ongelmat, joissa deterministinen algoritmi ei tahdo toimia, ovat helpohkosti ratkaistavissa "kolikkoa heittämällä"
- Satunnaisuus voi johtaa vinoumaan
- Kuitenkin eksaktin ratkaisun laskeminen voi olla liikaa aikaa vaativaa
- Esimerkiksi tilastotieteessä suosittu *otantamenetelmä*
  - Sen sijaan, että äänestyskäyttäytyminen yritettäisiin kysyä kaikilta äänioikeutetuilta, ennuste perustetaan satunnaisesti valitun populaation antamiin vastauksiin

## Luokka BPP

- Probabilistinen Turingin kone  $N$  on epädeterministinen kone, jossa jokainen epädeterministinen askel on kolikonheitto-askel
- Tällaisella askelella on kaksi mahdollista seuraajaa
- Kuhunkin laskentapuun haaraan  $b$  liitetään todennäköisyys

$$\Pr[b] = 2^{-k}$$

missä  $k$  on haaraan liittyvien kolikonheittoaskelten lukumäärä (syötteellä  $w$ )

- Todennäköisyys, että  $N$  hyväksyy syöteen  $w$  on

$$\Pr[N \text{ hyväksyy syöteen } w] = \sum_{b \in A} \Pr[b]$$

missä  $A$  on hyväksyvien laskentahaarojen joukko

- $\Pr[N \text{ hylkää syötteen } w] = 1 - \Pr[N \text{ hyväksyy syötteen } w]$
- Tavalliseen tapaan Turingin kone tunnistaa kielen, jos se hyväksyy kieleen kuuluvat merkkijonot ja hylkää ne, jotka eivät siihen kuulu
- Paitsi, että probabilistiselle koneelle sallitaan pieni virheen todennäköisyys
- Kaikilla  $0 \leq \epsilon < \frac{1}{2}$  sanomme, että  $N$  tunnistaa kielen  $A$  virhetodennäköisyydellä  $\epsilon$  jos
  - $w \in A \Rightarrow \Pr[N \text{ hyväksyy } w:n] \geq 1 - \epsilon$
  - $w \notin A \Rightarrow \Pr[N \text{ hylkää } w:n] \geq 1 - \epsilon$
- $N$ :ää simuloimalla saaavan ratkaisun virhe on siis korkeintaan  $\epsilon$
- Virheen todennäköisyys voi olla myös syötteen pituudesta riippuva; esim. eksponentiaalisen pieni:  $\epsilon = 2^{-n}$

**Määritelmä.** *BPP on niiden kielten luokka, jotka voidaan tunnistaa probabilistisella polynomi aikaisella Turingin koneella virhetodennäköisyydellä  $\frac{1}{3}$ .*

- Virhetodennäköisyydeksi kelpaisi  $\frac{1}{3}$ :n sijaan mikä tahansa vakio väliltä  $]0, \frac{1}{2}[$
- Seuraavan *vahvistuslemman* (amplification lemma) perusteella virhetodennäköisyys voidaan aina ajaa eksponentiaalisen pieneksi
- Jos satunnaisalgoritmin virhetodennäköisyys on  $2^{-100}$ , niin se antaa virheellisen vastauksen paljon todennäköisemmin laitteisto-  
virheen takia kuin siksi, että kolikonheitto on ollut epäsuotuisa.

**Lemma 10.5.** Olkoon  $\varepsilon$  kiinnitetty vakio aidosti välillä  $0$  ja  $\frac{1}{2}$ . Tällöin mille tahansa polynomille  $p(n)$ , virhetodennäköisyyden  $\varepsilon$  puitteissa toimivalla probabilistisellä polynomiakaisella Turingin koneella  $N_1$  on olemassa ekvivalentti polynomiainen probabilistinen Turingin kone  $N_2$ , jonka virhetodennäköisyys on korkeintaan  $2^{-p(n)}$ .

**Todistus.** (Ajatus)  $N_2$  simuloi  $N_1$ :tä suorittamalla sitä polynomisen määrän kertoja ja suorittamalla enemmistöäänestyksen. Virheen todennäköisyys pienenee eksponentiaalisesti suorituskertojen lukumäärän mukaan. □

## Alkuluvut

- Olkoon  $\mathbb{Z}_p^+ = \{0, \dots, p-1\}$
- Jokainen kokonaisluku on ekvivalentti modulo  $p$  jonkin joukon  $\mathbb{Z}_p^+$  alkion kanssa

**Fermat'n pieni lause.** Jos  $p$  on alkuluku ja  $a \in \mathbb{Z}_p^+$ , niin

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

- Esimerkiksi  $2^{7-1} = 2^6 = 64$  ja  $64 \bmod 7 = 1$   
kun taas  $2^{6-1} = 2^5 = 32$  ja  $32 \bmod 6 = 2$   
joten  $6$  ei ole alkuluku
- Luvun  $6$  todetaan olevan yhdistetty ilman tekijöihin jakoa!



- Fermat'n pieni lause siis (melkein) antaa alkulukutestin
- Luvun  $p$  sanotaan selvittävän Fermat'n testin pisteessä  $a$ , jos  $a^{p-1} \equiv 1 \pmod{p}$
- Luku  $p$  on *pseudo-alkuluku*, jos se selvittää Fermat'n testin kaikilla itseään pienemmillä luvuilla  $a$  s.e.  $a$  ja  $p$  ovat keskenään alkulukuja
- Vain harvinaiset ns. *Carmichaelin luvut* ovat pseudo-alkulukuja olematta alkulukuja
- Jos luku ei ole pseudo-alkuluku, niin se selvittää Fermat'n testin korkeintaan puolessa pisteitä
- Näin ollen saamme helposti eksponentiaalisen virhetodennäköisyyden omaavan pseudo-alkulukujen tunnistus-algoritmin



#### Pseudo-alkuluku( $p$ )

1. Valitse satunnaiset  $a_1, \dots, a_k \in \mathbb{Z}_p^+$
2. Laske arvo  $a_i^{p-1} \pmod{p}$  kullakin  $i$
3. Jos kaikki lasketut arvot ovat **1 hyväksy**, muuten **hylkää**

- Jos  $p$  ei ole pseudo-alkuluku, niin se selvittää kunkin satunnaisesti valitun testin korkeintaan todennäköisyydellä  $\frac{1}{2}$
- Todennäköisyys selvittää kaikista  $k$ :sta testistä on siis  $2^{-k}$
- Algoritmin aikavaativuus on polynominen syötteen pituuden suhteen
- Vielä pitäisi päästä eroon Carmichaelin luvuista

- Luvulla 1 on täsmälleen neliöjuuret 1 ja -1 modulo mikä tahansa alkuluku  $p$
- Useilla yhdistetyillä luvuilla, varsinkin kaikilla Carmichaelin luvuilla, 1:llä on neljä tai useampia neliöjuuria
- Esimerkiksi  $\pm 1$  ja  $\pm 8$  ovat luvun 1 neljä neliöjuurta modulo 21
- Saamme luvun 1 neliöjuuren jos  $p$  selvittää Fermat'n testin pisteessä  $a$  sillä
  - nyt  $a^{p-1} \bmod p \equiv 1$ , joten
  - $a^{(p-1)/2} \bmod p$  on 1:n neliöjuuri
- Voimme toistuvasti puolittaa eksponentin niin kauan kuin se pysyy kokonaislukuna

### Alkuluku( $p$ )

% hyväksy = syöte  $p$  on alkuluku

1. Jos  $p$  on parillinen, niin **hyväksy** jos  $p = 2$ , muuten **hylkää**
2. Valitse satunnaiset  $a_1, \dots, a_k \in \mathbb{Z}_p^+$
3. Jokaiselle  $i \in \{1, \dots, k\}$ 
  - a) Laske  $a_i^{p-1} \bmod p$  ja **hylkää** jos erisuuri kuin 1
  - b) Olkoon  $p - 1 = st$ , missä  $s$  on pariton ja  $t = 2^h$  on 2:n potenssi
  - c) Laske arvot  $a_i^{s \cdot 2^0}, a_i^{s \cdot 2^1}, \dots, a_i^{s \cdot 2^h}$  modulo  $p$
  - d) Jos jokin lasketuista arvoista on erisuuri kuin 1, niin valitse viimeinen niistä ja **hylkää**, jos se on erisuuri kuin -1
4. Kaikki testit on selvitetty, joten **hyväksy**

**Lemma 10.7** Jos  $p$  on pariton alkuluku, niin

$$\Pr[\text{Alkuluku hyväksyy } p:n] = 1.$$

**Todistus.** Jos  $p$  on alkuluku, niin yksikään laskentahaara ei hylkää: Hylkääminen askelessa 3a tarkoittaa, että  $(a^{p-1} \bmod p) \neq 1$ , joten Fermat'n pienen lauseen mukaan  $p$  on yhdistetty luku.

Jos hylkäys tapahtuu askelessa 3d, niin on olemassa  $b \in \mathbb{Z}_p^+$  s.e.

$$b \not\equiv \pm 1 \pmod{p} \text{ ja } b^2 \equiv 1 \pmod{p}. \text{ Näin ollen, } b^2 - 1 \equiv 0 \pmod{p}.$$

Tekijöimällä saadaan  $(b-1)(b+1) \equiv 0 \pmod{p}$ , jonka perusteella  $(b-1)(b+1) = cp$  jollakin positiivisella kokonaisluvulla  $c$ .

Koska  $b \not\equiv \pm 1 \pmod{p}$ , niin sekä  $b-1$  ja  $b+1$  ovat välillä  $]0, p[$ .

Täten  $p$  on yhdistetty luku, koska alkuluvun monikertaa ei voi ilmaista kahden sitä pienemmän luvun tulona.  $\square$

- Seuraava lemma osoittaa, että algoritmi tunnistaa yhdistetyt luvut suurella todennäköisyydellä
- Lukuteorian yhden perustuloksen, *kiinalaisen jäännöslauseen* (Chinese remainder theorem), mukaan on  $\mathbb{Z}_{pq}$ :n ja  $(\mathbb{Z}_p \times \mathbb{Z}_q)$ :n välillä on suora yhteys jos  $p$  ja  $q$  ovat keskenään alkulukuja:
  - Jokainen  $r \in \mathbb{Z}_{pq}$  vastaa paria  $(a, b)$ , missä  $a \in \mathbb{Z}_p$  ja  $b \in \mathbb{Z}_q$  s.e.
    - $r \equiv a \pmod{p}$  ja
    - $r \equiv b \pmod{q}$

**Lemma 10.8** Jos  $p$  on pariton yhdistetty luku, niin  
 $\Pr[\text{Alkuluku hyväksyy } p:n] \leq 2^{-k}$ .

**Todistus.** Sivutetaan, hyödyntää kiinalaista jäännöslausetta.  $\square$

- Olkoon  $\text{ALKULUVUT} = \{ n \mid n \text{ on alkuluvun binääriesitys} \}$
- Edellinen algoritmi ja sen analyysi antaa meille tuloksen

**Lause 10.9**  $\text{ALKULUVUT} \in \text{BPP}$

Huomattakoon, että alkulukujen satunnaisten tunnistaja-algoritmin virhe on *yksipuolinen*. Kun syöte hylätään, niin se on varmasti yhdistetty luku. Virhe voi tapahtua vain syöte hyväksyttäessä.

- Virhe tapahtuu siis vain yhdistetyllä luvulla. Kaikilla alkuluvuilla algoritmi antaa oikean vastauksen.
- Yksisuuntainen virhe on niin yleinen satunnaisalgoritmien ominaisuus, että sille on olemassa oma vaativuusluokkansa  $\text{RP}$

**Määritelmä 10.10**  $\text{RP}$  on niiden kielten luokka, jotka ovat tunnistettavissa polynomiaikaisella probabilistisellä Turingin koneella s.e. kieleen kuuluvat merkkijonot hyväksytään vähintään todennäköisyydellä  $\frac{1}{2}$  ja kieleen kuulumattomat hylätään todennäköisyydellä 1.

- Edellä olevan perusteella siis  $\text{YHDISTETYT} \in \text{RP}$

## ALKULUVUT $\in P$

- Fermat'n pienen lauseen seurauksena pätee:

**Lause A.** Olkoot  $a$  ja  $p$  keskenään alkulukuja ja  $p > 1$ .  $p$  on alkuluku jos ja vain jos  $(x - a)^p \equiv (x^p - a) \pmod{p}$

- $x$ :llä ei tässä ole merkitystä, vain polynomin  $(x - a)^p$  kertoimet merkitsevät
- Polynomin  $(p - 1)$ :n keskimäisen termin kertoimet ovat  $p$ :llä jaollisia kokonaislukuja, eli  $= 0$  modulo  $p$
- Jäljelle jää siis vain ensimmäinen termi  $x^p$  ja viimeinen termi  $-a^p$ , joka on  $-a$  modulo  $p$
- Valitettavasti vain  $p$ :n alkulukuominaisuuden ratkaiseminen tämän perusteella vaatii eksponentiaalisen ajan

- Agrawal (1999): riittää tarkastella polynomia  $(x - a)^p$  modulo  $x^r - 1$
- Jos  $r$  on riittävän suuri, ainoat testin läpäisevät yhdistetyt luvut ovat parittomien alkulukujen potenssit
- Luvun  $r$  siis tulisi olla riittävän suuri, mutta toisaalta riittävän pieni, jottei menetelmän vaativuus kasva liikaa
- Kayal & Saxena (2000): Todistamattomaan konjektuuriin perustuen  $r$ :n ei tarvitse olla suurempi kuin  $4(\log^2 p)$ , jolloin testimenetelmän vaativuus on vain  $O(\log^3 n)$  eli kuuluu  $P$ :hen
- Valitettavasti vain tulos perustuu todistamattomaan väitteeseen

- Sophie Germain –alkuluvut: sekä  $q$  että  $2q + 1$  ovat alkulukuja
- Agrawal, Kayal & Saxena (2002): Jos löytyy SG-alkulukujen pari  $q$  ja  $2q + 1$  s.e.

$$q > 4\left(\sqrt{2q+1}\right) \cdot \log p$$

niin  $r$ :n ei tarvitse olla arvoltaan kuin korkeintaan

$$2\left(\sqrt{2q+1}\right) \cdot \log p$$

- Valitettavasti tämä testi on rekursiivinen ja sen vaativuus on  $O(\log^{12} n)$  edellä mainitun  $O(\log^3 n)$ :n sijaan

### Deterministinen-alkuluku( $n$ )

1. if  $n$  on muotoa  $a^b$  jollain  $b > 1$  then hylkää;
2.  $r \leftarrow 2$ ;
3. while  $r < n$  do
  - a) if  $\text{syt}(n, r) \neq 1$  then hylkää;
  - b) if Deterministinen-alkuluku( $r$ ) then %  $r > 2$ 
    - i. Olkoon  $q$  ( $r-1$ ):n suurin tekijä;
    - ii. if  $q > 4\text{sqrt}(r) \log n$  and  $n^{(r-1)/q} \not\equiv 1 \pmod{r}$  then break;
  - c)  $r \leftarrow r + 1$ ;
4. for  $a \leftarrow 1$  to  $2\text{sqrt}(r) \log n$  do
  - if  $(x-a)^n \not\equiv (x^n-a) \pmod{x^r-1, n}$  then hylkää;
5. hyväksy syöte;

- Rivin 1 testi poistaa Agrawalin (1999) testin edellytysten mukaisesti parittomien alkulukujen potenssit
- Rivin 3 silmukka etsii Sophie Germain –alkulukuparin  $q$  ja  $r$
- Rivi 3a) testaa lausetta  $A$  varten, että  $n$  ja  $r$  ovat keskenään alkulukuja
- Rivin 4 silmukka tutkii alkulukuominaisuuden lauseen  $A$  muunnoksella (Agrawal, 1999) arvoon  $2\sqrt{r} \log n$  asti (AKS, 2002)
- Koska lause  $A$  pätee jos ja vain jos  $n$  on alkuluku, niin algoritmin antama päätös on korrekti
- Muut muutokset vaikuttavat vain algoritmin vaativuuteen, ei sen oikeellisuuteen

## Yhteenveto

- Kaikille laskennallisille ongelmille ei ole ohjelmallista ratkaisua
- Deterministisellä äärellisellä automaatilla on yksikäsitteinen minimiautomaatti. Minimiautomaatin muodostaminen on suoraviivaista
- Epädeterministiset äärelliset automaattit ovat yhtä vahvoja kuin deterministisetkin
- Kieli on säännöllinen
  - ⇔ se voidaan tunnistaa äärellisellä automaatilla
  - ⇔ se voidaan kuvata säännöllisellä lausekkeella
  - ⇔ se voidaan tuottaa oikealle lineaarisella kieliopilla



- Säännöllisen kielen merkkijonoja voidaan "pumpata"
- On olemassa mielekkäitä formaaleja kieliä, jotka eivät ole säännöllisiä
- Kontekstittomat kielet ovat säännöllisten kielten aito yli luokka
- Kieli on kontekstiton jos ja vain jos se voidaan tunnistaa pinoautomaatilla
- Churchin-Turingin teesin mukaan mikä tahansa tietokoneella ratkeava ongelma voidaan ratkaista Turingin koneella
- Turingin koneiden laajennukset — mukaan lukien epädeterministiset Turingin koneet — ovat laskentavoimaltaan ekvivalenteja standardimallisen yksinauhaisen koneen kanssa



- Eri konevarianttien "tehokkuudet" vaihtelevat
  - Rajoittamattomilla kielioppeilla tuotettavat kielet ovat ekvivalenteja Turingin koneilla tunnistettavien kielten kanssa
  - Totaalinen Turingin kone pysähtyy kaikilla syötteillä
  - Formaali kieli on RE-kieli, jos se voidaan tunnistaa Turingin koneella ja rekursiivinen, jos on olemassa totaalinen tunnistava kone
- $A$  ja  $B$  rekursiivisia  $\Leftrightarrow \bar{A}$ ,  $A \cup B$  ja  $A \cap B$  rekursiivisia
  - $A, B \in RE \Leftrightarrow (A \cup B), (A \cap B) \in RE$
  - $A$  rekursiivinen  $\Leftrightarrow A, \bar{A} \in RE$
  - $A \in RE$ , ei rekursiivinen  $\Leftrightarrow \bar{A} \notin RE$

- $D = \{c \in \{0, 1\}^* \mid c \notin L(M_c)\} \notin RE$
- Esim.  $A_{DFA}$ ,  $E_{DFA}$  ja  $EQ_{DFA}$  ovat ratkeavia (rekursiivisia) kieliä
- $U = \{\langle M, w \rangle \mid w \in L(M)\} \in RE$ , ei rekursiivinen
- $\bar{U} = \{\langle M, w \rangle \mid w \notin L(M)\} \notin RE$
- $H = \{\langle M, w \rangle \mid M(w) \downarrow\} \in RE$ , ei rekursiivinen
- $\bar{H} = \{\langle M, w \rangle \mid M(w) \uparrow\} \notin RE$
- Chomskyn kieliluokat:  
äärelliset  $\subsetneq$  säännölliset  $\subsetneq$  kontekstittomat  $\subsetneq$  kontekstiset  $\subsetneq$   
rajoittamattomilla kielioppeilla tuotettavat kielet (= RE)

- $NE = \{\langle M \rangle \mid M \text{ on Turingin kone ja } L(M) \neq \emptyset\} \in RE$ , ei rekursiivinen
- $REG = \{\langle M \rangle \mid M \text{ on Turingin kone ja } L(M) \text{ on säännöllinen}\}$  ei ole rekursiivinen
- Ricen lause: kaikki Turingin koneiden epätriviaalit semanttiset ominaisuudet ovat ratkeamattomia
- Lineaarisesti rajoitettu automaatti (LRA) ei voi ottaa lisää työtilaa käyttöönsä
- $A_{LRA}$  on ratkeava, kun taas  $E_{LRA}$  on ratkeamaton
- $A \subseteq \Sigma^*$  voidaan palauttaa rekursiivisesti  $B$ :hen,  $B \subseteq \Gamma^*$ , merk.  
 $A \leq_m B$ , jos on olemassa rekursiivinen funktio  $f: \Sigma^* \rightarrow \Gamma^*$  s.e.  
$$x \in A \Leftrightarrow f(x) \in B \quad \forall x \in \Sigma^*$$



- $A \subseteq \{0, 1\}^*$  on RE-täydellinen, jos
  1.  $A \in RE$  ja
  2.  $B \leq_m A$  kaikilla  $B \in RE$
- Kieli  $U$  on RE-täydellinen
- Turingin koneiden vaativuusanalyysissä tarkastellaan pahinta tapausta tietyn pituisella syötteellä
- Vaativuusfunktioiden kertaluokat  $O, \Theta, o, \Omega$
- Työnauhojen lukumäärä ei vaikuta oleellisesti Turingin koneen tehokkuuteen
- Deterministisen ja epädeterministisen koneen tehokkuusero sen sijaan on eksponentiaalinen



$$P = \bigcup_{k \geq 0} \text{DTIME}(n^k + k)$$

$$\text{EXPTIME} = \bigcup_{k \geq 0} \text{DTIME}(2^{n^k})$$

- $P$ :hen kuuluvat kielet, jotka voidaan tunnistaa syötteen pituuden suhteen polynomisessa ajassa
- Esim. suunnatun verkon polkuongelma PATH ja keskenään alkulukuja olevien lukuparien tunnistaminen ovat luokan  $P$  ongelmia
- Vastaavat epädeterministiset yhdisteluokat ovat  $NP$  ja  $NEXPTIME$
- Esim. klikkiongelma ja osajoukkosumma ovat luokan  $NP$  ongelmia
- Luokan  $P$  ongelmat ovat käytännössä ratkeavia

- $P \subseteq NP$ . Lisäksi  $NP$ :ssä on ongelmia, joille ei tunneta polynomista ratkaisualgoritmia
- $A \subseteq \Sigma^*$  voidaan palauttaa polynomisesti  $B$ :hen,  $B \subseteq \Gamma^*$ , merk.

$A \leq_m^p B$ , jos on olemassa polynomisessa ajassa laskettava funktio  $f: \Sigma^* \rightarrow \Gamma^*$  s.e.

$$x \in A \Leftrightarrow f(x) \in B \quad \forall x \in \Sigma^*$$

- $A \subseteq \{0, 1\}^*$  on  $NP$ -täydellinen, jos
  1.  $A \in NP$  ja
  2.  $B \leq_m^p A$  kaikilla  $B \in NP$
- Kaikki luokan  $NP$  ongelmat voidaan palauttaa  $NP$ -täydelliseen ongelmaan polynomisesti.
- Jos jokin  $NP$ -täydellinen ongelma kuuluu  $P$ :hen, niin  $P = NP$

- Ongelman  $A \in NP$  todistaminen  $NP$ -täydelliseksi:
  1. Valitse samankaltainen tunnetusti  $NP$ -täydellinen ongelma  $B$
  2. Muodosta polynominen palautus  $f: B \leq_m^p A$ ; lauseen 7.36 perusteella myös  $A$  on  $NP$ -täydellinen
- $NP$ -täydellisiä ongelmia: SAT, CSAT, 3SAT, VC, IS, KLIKKI, Hamiltonin polku, TSP, Osajoukkosumma ja minSC

$$\begin{aligned} PSPACE &= \bigcup_{k \geq 0} DSPACE(n^k) \\ NPSPACE &= \bigcup_{k \geq 0} NSPACE(n^k) \end{aligned}$$

$$P \subseteq NP \subseteq \left\{ \begin{array}{l} PSPACE = \\ NPSPACE \end{array} \right\} \subseteq EXPTIME \subseteq$$

$$NEXPTIME \subseteq \left\{ \begin{array}{l} EXPSPACE = \\ NEXPSPACE \end{array} \right.$$

- TQBF on PSPACE-täydellinen
- Samoin shakin ja GOn "asymptoottiset" versiot

$$L = DSPACE(\log n)$$

$$NL = NSPACE(\log n)$$

- $PATH \in NL$  on NL-täydellinen
- $L \stackrel{?}{=} NL$
- $NL \subseteq P$
- Laskennallisesti vaativaa ongelmaa voidaan pyrkiä approksimoimaan tehokkaasti
- Solmupeiteongelmalla on tehokas 2-approksimointialgoritmi
- Joukkopeiteongelmalla on tehokas  $(\ln m + 1)$ -approksimointialgoritmi

- Satunnaisalgoritmit ovat toinen mahdollinen tapa toimia laskennallisesti vaativien ongelmien kohdalla
- Alkulukujen tunnistaminen satunnaisalgoritmeilla on melko helppoa
- Nyttemmin on olemassa myös deterministinen polynomi aikainen algoritmi alkulukujen tunnistamiseksi