Probabilistiset algoritmit

- Viimeiseksi käsitellään probabilistisia algoritmeja.
- Tällaisessa algoritmissa arvataan tai "heitetään kolikkoa"ja tuloksen mukaan haaraudutaan.
- Erona epädeterminismiin on, ettei nyt tarkastella kaikkia haaroja, vaan laskenta toistetaan uusilla arvauksilla tarvittaessa. Lisäksi konkreettiset todennäköisyydet ovat usein esillä.
- Ehkä hieman yllättäen probabilistiset algoritmit ovat usein tehokkaampia kuin vastaavat deterministiset. Näin erityisesti kryptografisissa sovelluksissa.

Probabilistiset Turingin koneet

Määritelmä

Probabilistinen Turingin kone *M on eräänlainen epädeterminististen Turingin kone, jossa epädeterminististä askelta kutsutaan kolikon heitto -askeleeksi. Jokaiseen M:n haaraan b syötteellä w liitetään todennäköisyys*

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

missä k on haarassa b tapahtuvien kolikon heittojen lkm. Todennäköisyys, että M hyväksyy w:n, on

$$Pr[\mathrm{M\ \ accepts\ \ } w] = \sum_{b} Pr[b],$$

missä siis summa käy yli hyväksyvien haarojen.

Kielen hyväksyminen probabilistisella koneella

Määritelmä

M tunnistaa kielen A virhetodennäköisyydellä ε, $0 \leq \varepsilon < \frac{1}{2}$, *jos*

1. $w \in A$, *niin* $Pr[M \text{ accepts } w] \geq 1 - \varepsilon$ *ja jos*
2. $w \notin A$, *niin* $Pr[M \text{ rejects } w] \geq 1 - \varepsilon$.

- Siis väärän vastauksen todennököisyys on korkeintaan ε. Toistamalla algoritmia voidaan virhemahdollisuutta pienentää.
- Virhetodennäköisyys voidaan ilmoittaa usein myös syötteen pituuden funktiona, $\varepsilon = 2^{-n}$.

Määritelmä

BPP on niiden kielten luokka, jotka voidaan tunnistaa probabilistisella Turingin koneella polynomisessa ajassa virhetodennäköisyydellä $\frac{1}{3}$.

- Määritelmässä käytetään virhetodennäköisyyttä $\frac{1}{3}$, mutta mikä tahansa arvo väliltä $]0, \frac{1}{2}[$ kävisi yhtä hyvin seuraavan vahvistuslemman nojalla.
- Sen nojalla voidaan virhetodennäköisyys tehdä eksponentiaalisen pieneksi.

Vahvistuslemma

Lause

Olkoon ε mielivaltainen vakio väliltä $]0, \frac{1}{2}[$. Tällöin jokaista polynomisessa ajassa $p(n)$ toimivaa probabilistista Turingin konetta M_1, jonka virhetodennäköisyys on ε, kohti on olemassa ekvivalentti probabilistinen polynomisessa ajassa toimiva Turingin kone M_2, jonka virhetodennäköisyys on $2^{-p(n)}$.

Todistus.

- On siis annettu kone M_1, joka tunnistaa kielen virhetodennäköisyydellä ε, ja polynomi $p(n)$.
- Konstruoidaan kone M_2, joka tunnistaa saman kielen virhetodennäköisyydellä $2^{-p(n)}$.
- M_2 toimii syötteellä w seuraavasti:
  1. *Laske k (kts. todistuksen loppua).*
  2. *Simuloi M_1:tä $2k$ kertaa syötteellä w.*
  3. *Jos useimmat M_1:n simuloinnit hyväksyvät, hyväksy; muuten hylkää.*
- Arvioidaan yläraja todennäköisyydelle, että M_2 antaa väärän vastauksen syötteellä w.
- Askeleen perusteella saadaan $2k$ tulosta. Jokainen tulos on joko oikea tai väärä.
- Jos useimmat näistä tuloksista ovat oikeita, M_2 antaa oikean vastauksen.
- Arvioidaan tn, että vähintään puolet vastauksista ovat vääriä.
- Olkoon S sarja vastuksia, jotka voitaisiin saada askeleesta 2.

- Olkoon p_S tn, että M_2 tuottaa juuri S:n.
- Oletetaan, että S:ssä on c oikeaa ja w väärää vastausta, $c + w = 2k$.
- Jos $c \leq w$ ja M_2 laskee S:n, niin silloin M_2 tulostaa väärän vastauksen.
- Kutsutaan tällaista S:ää huonoksi jonoksi.
- Jos S on huono jono, niin $p_S \leq \varepsilon^w (1-\varepsilon)^c$, mikä puolestaan on korkeintaan $\varepsilon^k (1-\varepsilon)^k$, koska $k \leq w$ ja $\varepsilon < 1 - \varepsilon$.
- Summaamalla p_S:t kaikilla pahoilla joinoilla S antaa todennäköisyyden, että M_2 antaa väärän tuloksen.
- Pahoja jonoja on korkeintaan 2^{2k}, koska tämä on kaikkien jonojen lkm.
- Siten tn, että M_2 tulostaa väärän vastauksen syötteellä w, on

$$\sum_{\text{bad } S} p_S \leq 2^{2k} \cdot \varepsilon^k (1-\varepsilon)^k = (4\varepsilon(1-\varepsilon))^k.$$

- Oletettiin, että $\varepsilon < \frac{1}{2}$, joten $4\varepsilon(1-\varepsilon) < 1$. Siten yllä esitetty tn pienenee eksponentiaalisesti k:n suhteen ja samoin M_2:n virhetn.

- Arvioidaan lopuksi k, jolla M_2:n virhetodennäköisyys on alle 2^{-t} kaikilla $t \geq 1$.
- Olkoon $\alpha = \log_2(4\varepsilon(1-\varepsilon))$ ja valitaan $k \geq t/\alpha$.
- Tällöin virhetn on $2^{-p(n)}$. $\square$

Jaollisuus

- Alkuluku (prime) on sellainen ykköstä suurempi luonnollinen luku, joka on jaollinen vain yhdellä ja itsellään.
- Yhdistetty luku on sellainen ykköstä suurempi luku, joka ei ole alkuluku.
- Pitkään oli epäselvää, voidaanko alkulukutesti tehdä polynomisesti. Tämä tulos riippui laajennetusta Riemannin hypoteesista.
- Vuonna 2002 intialaiset keksivät kuitenkin suhteellisen yksinkertaisen polynomisen algoritmin.
- Kuitenkin edelleen tilanne lienee se, että probabilistinen alkulukutesti on tehokkain.

Käsitteitä

- Kokonaisluvut x ja y ovat ekvivalentteja modulo kokonaisluku p, jos $x - y = cp$ jollakin kokonaisluvulla c. Tällöin merkitään

$$x \equiv y \pmod{p}.$$

- $x \mod p$ on pienin luonnollinen luku y, jolla $x \equiv y \pmod{p}$.
- $\mathbf{Z}_p = \{0, 1, ..., p-1\}$, $\mathbf{Z}_p^* = \{1, ..., p-1\}$.

Lause (Fermat'n pieni lause)

Jos p on alkuluku ja $a \in \mathbf{Z}_p \backslash \{0\}$, niin

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

Fermat'n testi

- Fermat'n lausetta voidaan käyttää testattaessa, onko luku alkuluku.
- Nimittäin kaikilla a, $1 \leq a < p$, pätee $a^{p-1} \equiv 1 \pmod{p}$.
- Toisaalta jos p ei ole alkuluku, niin joillakin a:n arvoilla em. yhtälö ei ole voimassa.
- Esim. $p = 6$,

$$2^{(6-1)} = 2^5 = 32$$

  ja $32 \mod 6 = 2$.
- Kokonaisluku q on pseudoalkuluku, jos se läpäisee Fermat'n testin kaikilla $a < q$, joilla ei ole yhteisiä tekijöitä q:n kanssa.
- Pseudoalkuluvut ovat lähes sama asia kuin alkuluvut lukuunottamatta ns. Carmichaelin lukuja, jotka ovat yhdistettyjä, mutta silti läpäisevät Fermat'n testin.

Fermat'n testi II

- Seuraava algoritmi testaa, onko luku alkuluku vai ei. Se toimii oikein kaikissa muissa tapauksissa paitsi Carmichaelin lukujen yhteydessä.
- Jos luku ei ole pseudoalkuluku, se epäonnistuu puolessa tapauksista.

Algoritmi:

1. Valitse satunnaisesti $a_1, ..., a_k$ joukosta $1, ..., p-1$.
2. Laske $a_i^{p-1} \mod p$ kaikilla i.
3. Jos kaikki lasketut arvot ovat 1, hyväksy, muuten hylkää.

Fermat'n testi III

- Jos p on alkuluku, se läpäisee kaikki testit ja algoritmi hyväksyy p:n alkuluvuksi varmuudella.
- Jos p ei ole pseudoalkuluku, se läpäisee korkeintaa puolet testeistä. Tässä tapauksessa se siis läpäisee testin tietyn luvun kohdalla tn:llä $\frac{1}{2}$.
- Siten tn, että se läpäisee kaikki testit, on korkeintaan 2^{-k}.
- Algoritmi toimii polynomisessa ajassa, koska potensiinkorotus voidaan tehdä polynomisessa ajassa.
- Testi kuitenkin epäonnistuu Carmichaelin lukujen yhteydessä, joten sitä täytyy edelleen kehittää.

- Luvulla 1 on kaksi neliöjuurta 1 ja -1, modulo mikä tahansa alkuluku p.
- Monilla yhdistetyillä luvuilla, mukaan lukien Carmichaelin luvut, ykkösellä on neliöjuuria neljä tai useampia. Esim. 1, -1, 8 ja -8 ovat ykkösen neliöjuuria modulo 21.
- Jos luku läpäisee Fermat'n testin a:ssa, modifioitu algoritmi laskee ykkösen neliöjuuria modulo p. Jos jokin neliöjuuri ei ole 1 tai -1, niin p ei ole alkuluku.
- Jos p läpäisee Fermat'n testin a:ssa, niin $a^{(p-1)/2} \mod p$ on ykkösen neliöjuuri.
- Jos se on 1, eksponenttia voidaan jakaa kakkosella niin kauan, kun tulos kokonaisluku. Jokaisen jaon jälkeen lasketaan tulos ja jos se on eri kuin 1 tai -1, niin p ei ole alkuluku.

Alkulukutesti

Syöte on luku p:

1. Jos p on parillinen, hyväksy jos $p = 2$, muuten hylkää.
2. Valitse satunnaisesti a_1,...,a_k väliltä $[2, p-1]$.
3. **For** $i := 1$ to k **do**
4. Laske a_i^{p-1} mod p ja hylkää, jos tulos eri kuin 1.
5. Olkoon $p - 1 = st$, missä s on pariton ja $t = 2^h$.
6. Laske jono $a_i^{s \cdot 2^0}$, $a_i^{s \cdot 2^1}$,...,$a_i^{s \cdot 2^h}$ mod p.
7. Jos jonossa jokin luku ei ole 1, etsi viimeinen alkio $e \neq 1$ ja hylkää, jos $e \neq -1$.
8. Jos on päästy tähän asti, hyväksy.

Todistus I

Algoritmin oikeellisuustodistus perustuu kahteen lemmaan.

Lemma

Jos p on pariton alkuluku, niin tn, että algoritmi hyväksyy, on 1.

Todistus.

- Sanotaan, että luku a_i on todistaja, jos algoritmi päätyy hylkäävään tilaan askeleessa 4 tai 7 a_i:tä tutkiessaan.
- Osoitetaan ensin, että jos p on alkuluku, todistajia ei ole olemassa, joten algoritmi ei päädy hylkäävään tilaan.
- Jos a olisi todistaja askeleessa 4, niin

$$(a^{p-1} \mod p) \neq 1$$

ja Fermat'n pieni lause sanoo, että p on yhdistetty luku.

Todistus II

- Jos a on todistaja askeleessa 7, niin on olemassa $b \in \mathbf{Z}_p$, jolla $b \not\equiv \pm 1 \pmod{p}$ ja $b^2 \equiv 1 \pmod{p}$.
- Siten $b^2 - 1 \equiv 0 \pmod{p}$.
- Jakamalla $b^2 - 1$ tekijöihin saadaan

$$(b-1)(b+1) \equiv 0 \pmod{p},$$

josta seuraa

$$(b-1)(b+1) = cp$$

jollakin positiivisella luvulla c.

- Koska $b \not\equiv \pm 1 \pmod{p}$, sekä $b-1$ että $b+1$ ovat aidosti lukujen 0 ja p välissä.
- Siten p on yhdistetty luku, koska alkuluvun monikertaa ei voi ilmaista sitä pienempien lukujen tulona. $\square$

Kiinalainen jäännösteoreema

- Tarvitsemme kiinalaista jäännöslausetta, kun todistetaan, että algoritmi löytää yhdistetyt luvut riittävällä todennäköisyydellä.
- Kiinalainen jäännöslause sanoo, että jos p ja q ovat keskenään jaottomia, niin on olemassa bijektio lukujen $r \in \mathbf{Z}^*_{pq}$ ja sellaisten parien pari (a, b) välillä, että $a \in \mathbf{Z_p}$, $b \in \mathbf{Z_q}$ ja

$$\begin{aligned} r &\equiv a \pmod{p}, \\ r &\equiv b \pmod{q}. \end{aligned}$$

Lemma

Jos p on pariton yhdistetty luku, niin todennäköisyys, että algoritmi hyväksyy syötteen, on korkeintaan 2^{-k}.

Todistus.

- Näytetän, että jos p on pariton yhdistetty luku ja a valitaan satunnaisesti $\mathbf{Z}_p$:stä, niin

$$Pr[a \text{ witness}] \geq \frac{1}{2}.$$

- Tämä osoitetaan siten, että näytetään todistajia olevan vähintään yhtä paljon kuin ei-todistajia $\mathbf{Z}_p$:ssä. Tämä näytetään etsimällä jokaista ei-todistajaa kohti yksikäsitteinen todistaja.
- Jokaisen ei-todistajan yhteydessä askeleessa 6 saadaan jono, jossa joko kaikki alkiot ovat ykkösiä tai -1 on jossain paikassa ja sitä seuraa ykkösjono.

- Sanomme, että ei-todistaja on ensimmäistä tyyppiä, jos se tuottaa vain ykkösiä, ja toista tyyppiä, jos se tuottaa myös -1:n. Esim. 1 itse on ensimäistä tyyppiä, ja -1 toista tyyppiä.
- Toisen tyypin ei-todistajien joukosta otetaan ei-todistaja, jonka yhteydessä -1 esiintyy eniten oikealla vastaavassa jonossa.
- Olkoon h tuo ei-todistaja ja j potenssi, jonka yhteydessä -1 esiintyy. Eli $h^{s\cdot 2^j} \equiv -1 \pmod{p}$.
- Koska p on yhdistetty luku, niin joko p on alkuluvun potenssi tai se on tulo qr, missä r ja q ovat keskenään jaottomia.
- Tarkastellaan jälkimmäistä vaihtoehtoa ensiksi. Kiinalaisen jäännöslauseen mukaan on olemassa $t \in \mathbf{Z}_p$, jolla
  $t \equiv h \pmod{q}$,
  $t \equiv 1 \pmod{r}$
- Tällöin
  $t^{s\cdot 2^j} \equiv -1 \pmod{q}$,
  $t^{s\cdot 2^j} \equiv 1 \pmod{r}$

- Täten t on todistaja, koska $t^{s\cdot 2^j} \not\equiv \pm 1 \pmod{p}$, mutta $t^{s\cdot 2^{j+1}} \equiv 1 \pmod{p}$.
- Osoitetaan seuraavaksi, että $dt \mod p$ on yksikäsitteinen todistaja jokaiselle ei-todistajalle d.
- Ensiksikin $d^{s\cdot 2^j} \equiv \pm 1 \pmod{p}$ ja $d^{s\cdot 2^{j+1}} \equiv 1 \ (p)$ johtuen j:n valinnasta.
- Siten $dt \mod p$ on todistaja, koska $(dt)^{s\cdot j} \not\equiv \pm 1$ ja $(dt)^{s\cdot 2^{j+1}} \equiv 1 \pmod{p}$.
- Toiseksi jos d_1 ja d_2 ovat erillisiä ei-todistajia, niin

$$d_1 t \mod p \neq d_2 t \mod p.$$

- Siten $t \cdot t^{s\cdot 2^{j+1}-1} \mod p = 1$.
- Siksi jos $td_1 \mod p = td_2 \mod p$, niin

$$d_1 = t \cdot t^{s\cdot 2^{j+1}-1} d_1 \mod p = t \cdot t^{s\cdot 2^{j+1}-1} d_2 \mod p = d_2.$$

- Täten todistajien lkm on yhtä suuri kuin ei-todistajien, ja olemme analysoineet tapauksen, jossa p ei ole alkulukupotenssi.
- Toinen tapaus on tilanne $p = q^e$, missä q on alkuluku ja $e > 1$.
- Olkoon $t = 1 + q^{e-1}$. Binomiteoreeman mukaan

$$t^p = (1 + q^{e-1})^p = 1 + p \cdot q^{e-1} + \cdots ,$$

  joka on ekvivalenttia 1:n kanssa $\mod p$.
- Siten t on vaiheen 4 todistaja, koska tapauksessa $t^{p-1} \equiv 1 \pmod p$ pätee $t^p \equiv t \not\equiv 1 \pmod p$.
- kuten ensimmäisessä tapauksessa tätä todistajaa voidaan käyttää muiden todistajien löytämiseen.
- Jos d on ei-todistaja, niin $d^{p-1} \equiv 1 \pmod p$, mutta silloin dt mod p on todistaja.
- Lisäksi jos d_1 ja d_2 ovat erillisiä ei-todistajia, niin

$$d_1 t \mod p \neq d_2 t \mod p.$$

- Muuten

$$d_1 = d_1 \cdot t \cdot t^{p-1} \mod p = d_2 \cdot t \cdot t^{p-1} \mod p = d_2.$$

- Siten todistajien lkm on yhtä suuri kuin ei-todistajien ja todistus on viety loppuun. $\square$

Polynomien samuus I

- Tarkastellaan usean muuttujan polynomeja $p(x_1, ..., x_n)$.
- Tällaisen polynomin termin aste on sama kuin termin muuttujien eksponenttien summa. Polynomin aste on suurin termin aste.
- Jos p on annettu perusmuodossaan, on helppo selvittää, onko polynomi nollapolynomi vai ei: Tarkistetaan kaikkien termien kerroin, onko se nolla.
- Jos polynomi ei ole perusmuodossaan, ei tunneta polynomista algoritmia, joka testaisi, onko kyseessä nollapolynomi vai ei.
- Satunnaisalgoritmi nollapolynomin testaukseen perustuu seuraavaan lauseeseen.

Lause

Olkoon $p(x_1, ..., x_n)$ useamman muuttujan astetta d oleva polynomi. Jos p ei ole nollapolynomi, niin sellaisten juurien $(a_1, ..., a_n)$ lukumäärä, missä $-nd \leq a_i \leq nd$, on korkeintaan $nd(2nd + 1)^{n-1}$.

Polynomien samuus II

- Niiden alkioiden $(a_1, ..., a_n)$ lukumäärä, joilla $-nd \leq a_i \leq nd$, on $(2nd + 1)^n$.
- Siten tn, että polynomi saa arvon nolla ym alkiolla, on korkeintaan

$$\frac{nd(2nd+1)^{n-1}}{(2nd+1)^n} = \frac{1}{2+1/nd} < 1/2.$$

- Saadaan seuraava algoritmi nollapolynomin ja polynomien samuuden testaukseen. Syötteenä on usean muuttujan polynomi $p(x_1, ..., x_n)$:
  1. $d :=$ degree of p;
  2. **for** $i := 1$ **to** n **do** $a_i :=$ **random**$(-nd, nd)$;
  3. **if** $p(a_1, ..., a_n) \neq 0$ **then** *reject* **else** *accept*.
- Polynomin asteen laskeminen on triviaalia. Siten algoritmi toimii polynomisessa ajassa.

Pariutus verkossa

Määritelmä

Olkoon $G = (V, E)$ suuntaamaton verkko. Täydellinen pariutus (perfect matching) on sellainen osajoukko $E' \subseteq E$, että kaikilla verkon solmuilla u löytyy täsmälleen yksi $e \in E'$, johon u kuuluu.

Ongelmana on nyt selvittää, onko verkossa täydellistä pariutusta. Tämä voidaan tehdä edellisellä satunnaisalgoritmilla seuraavan lauseen nojalla.

Lause

Olkoon G suuntaamaton verkko, jonka solmujoukko on $\{1, ..., n\}$. Määritellään matriisi A kaavalla

$$a_{ij} = \begin{cases} x_{ij} & \text{jos } i \text{ on } j\text{:n vierussolmu ja } i < j, \\ -x_{ji} & \text{jos } i \text{ on } j\text{:n vierussolmu ja } i > j, \\ 0 & \text{muuten} \end{cases}$$

Tällöin G:ssä on täydellinen pariutus, jos ja vain jos A:n determinantti ei ole identtisesti nolla.

Todistus I

- A:n determinantti on määritelmän mukaan

$$\det A = \sum_{\pi} \sigma_{\pi} \prod_{i=1}^{n} a_{i\pi(i)},$$

missä π tarkoittaa joukon $\{1, ..., n\}$ permutaatiota ja σ_{π} on 1, jos π on tulo parillisesta määrästä vaihdoksia, -1, jos π on tulo parillisesta määrästä vaihdoksia.

- Kaikilla permutaatioilla π

$$\prod_{i=1}^{n} a_{i\pi(i)} \neq 0$$

jos ja vain jos i on $\pi(i)$:n vierussolmu eli permutaatio π vastaa G:n aliverkkoa G_{π}, joka koostuu särmistä $(i, \pi(i))$, $1 \leq i \leq n$.

Todistus II

- Tällainen aliverkko koostuu siis erilllisistä sykleistä, jotka peittävät kaikki G:n solmut.
- Huomataan ensiksi, että mikäli permutaatio π on sellainen, että G_π sisältää vähintään yhden parittoman syklin, niin se ei vaikuta determinantin arvoon.
- Tällaiset permutaatiot voidaan nimittäin ryhmitellä pareiksi, jotka kumoavat toisensa summassa seuraavaan tapaan.
- Permutaation π pariksi otetaan permutaatio π', joka on muuten sama kuin π, mutta pariton sykli on käännetty. Tällöin

$$\prod_{i=1}^{n} a_{i\pi(i)} = -\prod_{i=n}^{n} a_{i\pi'(i)}$$

ja $\sigma_\pi = \sigma_{\pi'}$, joten näiden permutaatioiden summa on nolla.

Todistus III

- Siten riittää tutkia permutaatioita, joita vastaavat verkot koostuvat parillisista sykleistä. Myös tällaiseen permutaatioon π voidaan liittää toinen permutaatio π^r, jossa kaikki syklit on käännetty.
- Edelleen $\sigma_\pi = \sigma_{\pi^r}$.
- Erotellaan kaksi tapausta:

  i) $\pi = \pi^r$. Tässä tapauksessa G_π koostuu kahden mittaisista sykleistä ja π vastaa täydellistä pariutusta E', jolla $\prod_{i=1}^{n} a_{i\pi(i)} = (t_{E'})^2$.
  (Tässä $t_{E'}$ tarkoittaa muuttujien x_{ij} tuloa, jotka vastaavat E':n kaaria.)

  ii) $\pi \neq \pi^r$. Tässä tapauksessa sekä π että π^r vastaavat kahta täydellistä pariutusta E' ja E'', jotka saadaan valitsemalla joka toinen särmä sykleistä. Tällöin

$$\prod_{i=1}^{n} a_{i\pi(i)} + \prod_{i=1}^{n} a_{i\pi^r(i)} = 2t_{E'}t_{E''}.$$

Todistus IV

- Determinantin arvo on siis sama kuin

$$(t_{E_1'} + t_{E_2'} + \cdots + t_{E_h'})^2$$

missä E_i' tarkoittaa i. täydellistä pariutusta. Summa on nolla vain, jos G:ssä ei ole täydellistä pariutusta. $\square$

Täydellisen pariutuksen olemassaolon algoritmi

- Nyt siis täydellisen pariutuksen ongelma on palautettu nollapolynomin testaukseen.
- Tämän ratkaisu oli helppoa satunnaisalgoritmilla.

Satunnaisluokkia

- Olemme siis osoittaneet että $\mathrm{PRIMES}, \mathrm{MATCHING} \in \mathrm{BPP}$.
- Kumpikin esittelemämme algoritmi voi erehtyä, mutta vain puoliksi.
- Jos alkulukutestissä tapahtuu hylkääminen, niin tiedämme varmasti, että luku on yhdistetty luku.
- Jos taas hyväksytään, niin luku voi olla yhdistetty tai alkuluku.
- Siten virhe voi tapahtua vain yhdistettyjen lukujen kohdalla.
- Vastaavalla tavalla jos polynomitestissä saamme tuloksen, että polynomi ei ole nollapolynomi, niin tulos on varma. Sen sijaan toteaminen nollapolynomiksi toteaminen voi sisältää virheen.
- Tarkennetaan nyt luokan BPP määritelmää.

Luokka RP

Määritelmä

RP on kieliluokka, jonka alkiot tunnistetaan probabilistisella, polynomisessa jassa toimivalla Turingin koneella seuraavasti. Jos alkio kuuluu kieleen, niin se tunnistetaan vähintään todennäköisyydellä $\frac{1}{2}$. Jos alkio ei kuulu kieleen, sen hylätään todennäköisyydellä 1.

- Esim. COMPOSITE on RP:n alkio.

Luokkien suhteita

Tiedetään seuraavat suhteet:

- $RP \cup coRP \subseteq BPP$.
- $RP \subseteq NP$ ja $coRP \subseteq coNP$.

Lisäksi voidaan määritellä uusia luokkia ottamalla käyttöön toisen tyyppisiä probabilistisia algoritmeja. Esim. Las Vegas -algoritmit antavat joko aina oikean vastauksen tai toisinaan vastauksen "en tiedä".

Luokissa RP ja BPP ei tunneta täydellisiä ongelmia ja on viitteitä, ettei sellaisia olekaan.