

$P \stackrel{?}{=} NP$

**Kysymys ratkaisun keksimisestä
ja sen tarkistamisesta**

Juha Nurmonen
Matematiikan laitos
Helsingin yliopisto

Ajatellaanpa esimerkiksi kauppamatkustajan jokapäiväistä ongelmaa: Kauppamatkustajan on käytävä matkallaan useassa kaupungissa myymässä tuotteitaan. Voiton maksimimiseksi hän haluaa luonnollisesti kiertää nämä kaupungit sopivan lyhyen – esimerkiksi enintään tuhannen kilometrin pituisen – reitin kautta, jotta matka olisi voittoa ja hän pääsisi pikaisesti kotiin nauttimaan tuotostaan. Mikäli kauppamatkustaja tietää tällaisen sopivan reitin – esimerkiksi lukemattomien myyntimatkojen kokemuksella tai onnistuneella arvauksella – on hänen valintansa matkasuunnitelmaksi helppo. Mutta jos hän ei sitä tiedä, saattaa sopivan reitin löytäminen osoittautua varsin työlääksi. Kauppamatkustaja ei nimittäin pahimmassa tapauksessa voi löytää sopivaa reittiä ennen kuin on käynyt läpi kaikki vaihtoehdot – näitä on esim. kymmenen kaupungin tapauksessa useampi miljoona! Tämä ongelma on esimerkki ratkaisun keksimisen ja sen tarkistamisen vaativuuksien eroista.

Yksi turhauttavampia teoreettisen tietojenkäsittelytieteen avoimia kysymyksiä viimeiset kolmekymmentä vuotta onkin ollut ns. $P \stackrel{?}{=} NP$ -kysymys. Kaikki mitä tietokoneella pystytään laskemaan, voidaan nimittäin aina esittää ns. päätösongelman muodossa: “onko annetulla syötteellä kysytty ominaisuus” (esimerkiksi kauppamatkustajan ongelman tapauksessa “onko olemassa reittiä, joka on lyhyempi kuin annettu luku”). Luokat P ja NP kuvaavat tällaisten päätösongelmien laskennallista eli algoritmista vaativuutta. Luokka P voidaan kuvailla eräänlaisena teoreettisena likiarvona niiden ongelmien kokoelmalle, joille voidaan järkevässä ajassa laskea ratkaisu. Luokka NP taas on kokoelma ongelmia, joille ratkaisun keksimisen – arvaamisen – jälkeen voidaan mielekkäässä ajassa tarkistaa ratkaisun oikeellisuus. Mainittakoon heti aluksi, että luokassa NP ovat edellä mainitun kauppamatkustajan ongelman lisäksi mm. seuraavat jokapäiväisessä verkkoteoriassa vastaantulevat ongelmat (joiden ei siis tiedetä olevan luokassa P):

- ☞ **Hamiltonisuus:** onko verkossa Hamiltonin sykliä; eli voidaanko verkossa kulkea verkon kaaria pitkin käymällä kussakin solmussa täsmälleen kerran ja palata alkupisteeseen;
- ☞ **3-väritettävyys:** voiko verkon värittää (korkeintaan) kolmella eri värillä siten, että toisiinsa yhteydessä olevat solmut ovat erivärisiä;
- ☞ **SPK:** onko verkossa klikkiä, joka sisältää puolet solmuista (klikki on solmujoukko, jossa kunkin kahden solmun välillä on yhteys).

Näillä ja monilla vastaavilla ongelmilla on myös tärkeä rooli useissa sovelluksissa käytetyissä algoritmeissa. Jos nimittäin niille löydettäisiin polynomiainainen ratkaisu, eli niiden tiedettäisiin olevan luokassa \mathbf{P} , monen tietokonepohjaisen sovelluksen uskottaisiin olevan tehokkaampi.

Tuntuisi aika luonnolliselta, että ratkaisun keksimisen laskennallinen vaativuus ja tämän ratkaisun oikeellisuuden tarkistamisen vaativuus ovat jotain aivan eri suuruusluokkaa. Mutta se, että $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ -kysymys on vielä kolmen vuosikymmenen tarmokkaiden yritysten jälkeen avoin, on sitäkin hämmästyttävämpää. Tosin kysymyksen ratkaisun osin varsin yllättävät seuraukset antanevat osviittaa kysymyksen syvällisyydestä ja ratkaisun vaikeudesta.

Mistä siis oikeastaan on kysymys?

A. Mitkä ihmeen \mathbf{P} ja \mathbf{NP} ?

$\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ -ongelma voidaan kaikessa lyhykäisyydessään määritellä kysymyksenä, voidaanko epädeterministisellä algoritmilla polynomisessa ajassa tunnistettava kieli tunnistaa myös deterministisellä algoritmilla polynomisessa ajassa. Tässä polynomiainaisuudella tarkoitetaan, että laskennan algoritmin askelten (kukin askel kestää yhden “aikayksikön”) määrä on rajoitettu jollakin tietyllä polynomilla syötteen pituuden suhteen.

Laskennan malli

Jotta tästä kysymyksenasettelusta saataisiin tarkka kuvaus, on ensin kuitenkin määriteltävä laskennan, tai tietokoneen, formaali malli. Standardi ja perinteinen tällainen käsite on *Turingin kone*, eräänlainen ideaalinen tietokone. Tämä laskennan malli esiteltiin jo paljon ennen ensimmäistäkin tietokonetta [Tur37]. Siitä huolimatta tällä “koneella” voidaan simuloida mitä tehokkaimpia tietokoneen ja laskennan malleja. On kuitenkin huomattava, että muitakin vastaavia laskennan malleja on olemassa, ja kaikki nämä mallit ovat oleellisesti ekvivalentteja. (Eri malleilla laskennan vaativuus muuttuu jonkin verran, muttei niin paljon, että sillä olisi merkitystä kysymyksenasetteluun.) Kuvailimme seuraavassa lyhyesti (jokseenkin teknisen) Turingin koneen määritelmän. (Jos lukija ei ole innostunut ao. teknisestä määritelmästä, hän voi myöhemmissä luvuissa huoletta korvata “Turingin koneen” “tietokoneella”.)

Olkoon annettu äärellinen *aakkosto* Σ , eli epätyhjä joukko symboleita, jossa on ainakin kaksi alkioa (aakkoston symboleille käytetään usein merkintää a, b , jne.); lisäksi on käytössä ns. tyhjä merkki $\lambda \notin \Sigma$. Turingin kone koostuu äärettömästä syötenauhasta, jossa kukin nauhapaikka voi sisältää arvonaan aakkoston Σ symbolin tai merkin λ . Alkutilanteessa nauhalla on jokin äärellinen määrä Σ :n merkkejä ja muissa nauhapaikoissa on tyhjä merkki λ .

Formaalisti Turingin kone M on monikko (Q, Σ, δ) , missä

- Q on koneen *tilojen* äärellinen joukko;
- Σ on koneen *nauha-aakkosto* ($\lambda \notin \Sigma$) ja
- $\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow Q \times (\Sigma \cup \{\lambda\}) \times \{\mathbf{R}, \mathbf{N}, \mathbf{L}\}$ on koneen *siirtymäfunktio*.

Tilojen joukko sisältää aina kolme erityistä tilaa

- q_0 on koneen *alkutila*;
- q_{hyv} on koneen *hyväksyvä* tila ja
- q_{hylk} on koneen *hylkäävä* tila.

Erityisesti kannattaa huomata, että δ kuvaa yksinkertaisesti äärellistä ohjelmaa. Tässä $\delta(q, m) = (q', m', f)$ vastaa sitä, että kone M on tilassa q ja nauhapää lukee syötenauhalla merkin m ja siirtymäfunktio δ siirtää koneen tilaan q' , kirjoittaa nauhalle (nauhapään kohdalle) symbolin m' ja siirtää nauhapään sitten pykälän vasemmalle, oikealle, tai jättää siirtämättä riippuen symbolista f , joka on joko L, R tai N.

Tämä yksinkertainen kone riittääkin kuvaamaan kaiken tarpeellisen. Erityisesti on huomattava, että laskenta ei välttämättä pysähdy lainkaan.

Luokat P ja NP

Luokat **P** ja **NP** ovat kielten kokoelmia: Olkoon Σ aakkosto. Joukon Σ äärellisille merkkijonoille käytetään merkintää Σ^* . Aakkoston Σ *kieli* onkin sitten yksinkertaisesti osajoukko $L \subseteq \Sigma^*$. Kuhunkin Turingin koneeseen M liittyy vastaavasti syöteaakkosto Σ . Samoin jokaista merkkijonoa $w \in \Sigma^*$ vastaa koneen laskenta tällä syötteellä w . Kone M *hyväksyy* merkkijonon (eli sanan) w , jos tämä laskenta pysähtyy hyväksyvään tilaan. Jos taas M pysähtyy hylkäävään tilaan tai laskenta ei pysähdy, M ei hyväksy jonoa w . Koneen M *hyväksymä kieli*, $L(M)$, on hyväksytyjen merkkijonojen joukko, eli

$$L(M) = \{w \in \Sigma^* \mid M \text{ hyväksyy jonon } w\}.$$

Olkoon sitten $t_M(w)$ koneen M käyttämien laskennan askelten määrä syötteellä w (jos laskenta ei koskaan pysähdy, on $t_M(w) = \infty$). Olkoon vielä $n \in \mathbb{N}$ ja

$$T_M(n) = \max\{t_M(w) \mid w \in \Sigma^n\},$$

missä Σ^n on n :n pituisten Σ :n merkkijonojen joukko. Tämä siis tarkoittaa, että mitaamme n :n pituisten jonojen laskennan pahimman tapauksen vaativuutta mittarilla $T_M(n)$. Kone M on polynomiainainen, jos jollakin $k \in \mathbb{N}$ pätee $T_M(n) \leq n^k + k$ kaikilla luonnollisilla luvuilla n . Luokan **P** formaali määritelmä kuuluu näiden esivalmistelujen jälkeen seuraavasti.

1 MÄÄRITELMÄ.

P on (jonkin aakkoston Σ) luokka

$$\mathbf{P} = \{L \mid L = L(M) \text{ jollakin polynomiainaisella koneella } M\}.$$

Luokka **NP** (nondeterministic polynomial time) määritellään perinteisesti vastaavasti kuten **P**, mutta käyttäen ns. epädeterminististä konetta, jossa tilasiirtymäfunktio voikin antaa kussakin tilassa useita arvoja (eli joukon) yhden sijaan. Toinen tapa määritellä tämä luokka on käyttää ns. *tarkistusrelaatiota*, joka on kaksipaikkainen relaatio $R \subseteq \Sigma^* \times \Sigma_1^*$, missä Σ ja Σ_1 ovat aakkostoja. Kuhunkin tällaiseen relaatioon R voidaan puolestaan liittää aakkoston $\Sigma \cup \Sigma_1 \cup \{\dagger\}$ (tässä $\dagger \notin \Sigma$) kieli L_R , joka määräytyy joukkona $L_R = \{w\dagger y \mid R(w, y)\}$. Relaatio R on polynomiainainen, jos sitä vastaava kieli on, eli jos $L_R \in \mathbf{P}$. Nyt voimme antaa luokan **NP** formaalin määritelmän.

2 MÄÄRITELMÄ.

Luokka **NP** on niiden (jonkin aakkoston Σ) kielten L luokka, joille löydetään jokin luonnollinen luku $k \in \mathbb{N}$ ja polynomiaikainen tarkistusrelaatio R , siten että kaikilla $w \in \Sigma^*$ pätee

$$w \in L \iff \exists y (|y| \leq |w|^k \text{ ja } R(w, y)),$$

missä $|w|$ ja $|y|$ merkitsevät näiden merkkijonojen pituutta.

Tässä siis tarkistusrelaatio R vastaa alussa mainitsemaamme löydettyä ratkaisua ja yllä oleva kaava onkin sitten vain polynomiaikainen tarkistus, onko tämä löytämämme ratkaisu oikea. Vastaavasti polynomisessa algoritmissa R vastaa “keksittyä” algoritmia, jonka oikeellisuus vain sitten todennetaan.

B. Mistä olikaan kysymys?

Nyt kun suurella vaivalla saimme tämän artikkelin pääluokat määriteltyä, voimme tarkastella niiden suhdetta. Ensinnäkin toteamme, että $\mathbf{P} \subseteq \mathbf{NP}$: jos (jonkin aakkoston Σ) kieli L on luokassa \mathbf{P} , saadaan kaivattu polynomiaikainen tarkistusrelaatio $R \subseteq \Sigma^* \cup \Sigma^*$ määrittelemällä $R(w, y) \iff w \in L$ kaikilla $w, y \in \Sigma^*$.

“Miljoonan taalan” kysymys onkin, onko tämä sisältyvyys aito!

Tokihan kullekin päätösongelmalle yo. määritelmän kaltainen ratkaisu löydetään, mutta tällainen triviaali, kaikki mahdollisuudet läpi käyvä ratkaisu, on pahimmasa tapauksessa väistämättä eksponentiaalinen (eli n :n pituisella syötteellä $T_M(n)$ on oleellisesti muotoa 2^{kn} jollakin vakiolla k). Ongelmana siis onkin, onko sille mahdollista löytää *polynomista* ratkaisua.

Tästä kysymyksestä tekevät erityisen mielenkiintoisen laskettavuuden teoriasta tutut palautuvuuden ja täydellisyyden käsitteet. Luonnollisesti myös luokkien **NP** ja **P** yhteydessä voidaan puhua aivan samoista käsitteistä.

3 MÄÄRITELMÄ.

- Tarkastellaan aakkoston Σ_1 kieltä L_1 ja aakkoston Σ_2 kieltä L_2 . Silloin L_1 on p -palautuva kieleen L_2 , eli $L_1 \leq_p L_2$, jos on olemassa polynomiaikaisesti laskettava funktio $f : \Sigma_1^* \rightarrow \Sigma_2^*$, jolle $x \in L_1 \iff f(x) \in L_2$ kaikilla $x \in \Sigma_1^*$.
- Kieli L on **NP**-täydellinen, jos $L \in \mathbf{NP}$ ja $L' \leq_p L$ kaikilla $L' \in \mathbf{NP}$.

Tästä saammekin suoraan tärkeimmät työkalut otsikon kysymyksen ratkaisemiseksi.

4 LAUSE.

Jos yksikin **NP**-täydellinen ongelma voidaan ratkaista polynomisessa ajassa, niin $\mathbf{P} = \mathbf{NP}$.

Ja sama kääntäen kuuluu tietysti seuraavasti.

5 SEURAUUS.

Jos $\mathbf{P} \neq \mathbf{NP}$, niin yhdelläkään **NP**-täydellisellä ongelmalla ei ole polynomiaikaista ratkaisualgoritmia.

Toisin sanoen kysymys näiden kahden *luokan* suhteesta palautuukin konkreettiseen kysymykseen yhdestä ainoasta (**NP**-täydellisestä) *ongelmasta!* Juuri tämä seikka on **NP**-täydellisyyden käsitteen tärkein arvo.

Ennen kuin tästä käsitteestä on tietenkään mitään iloa, on osoitettava, että on olemassa **NP**-täydellisiä ongelmia. Ensimmäisen tällaisen todisti Cook [Coo71]: ns. *ratkeavuusongelma* on **NP**-täydellinen. Ratkeavuusongelma on seuraavanlainen:

syöte: propositiolause;

kysymys: onko tämä propositiolause toteutuva.

Propositiolause koostuu propositiosymboleista p_0, \dots, p_k jollakin $k \in \mathbb{N}$, niiden negaatioista, sekä disjunktioista ja konjunktioista. Kysymys onkin siis määrittää, onko olemassa totuusjakaumaa v , joka antaa ko. lauseelle arvon tosi. Esimerkiksi propositiolauseelle $p_0 \wedge \neg p_0$ tällaista totuusjakaumaa ei löydy, mutta lauseelle $p_0 \wedge \neg p_1$ totuusjakauma $v(p_0) = \text{tosi}$ ja $v(p_1) = \text{epätosi}$ antaa lauseelle totuusarvon tosi. (Vaadittava polynomiaikainen tarkistusrelaatio Cookin tuloksessa on $R(x, y)$, joka on voimassa täsmälleen silloin, kun x on kyseisen propositiolauseen koodi ja y on propositiolauseen todeksi tekevän totuusjakauksen koodi.) Tässä kysymyksessä tulee erityisen hyvin esiin ratkaisun löytämisen ja sen tarkistamisen vaatuuksien ero: Jos on arvattu oikea totuusjakauma annetulle propositiolauseelle, on suoraviivaista ja laskennalliselta vaativuudelta varsin alhaista tarkistaa sen oikeellisuus. Mutta tällaisen totuusjakauksen löytäminen, kun kaikki mahdollisuudet on käytävä läpi, on vaativuudeltaan sen sijaan työlästä (muistammehan, että puhuimme pahimman tapauksen vaativuudesta).

Tänä päivänä **NP**-täydellisiä ongelmia tunnetaan ehkä jopa lähes tuhatkunta! Esimerkiksi teoksessa [GJ79] on lueteltu noin kolmesataa tällaista ongelmaa, joita joskus jopa kutsutaan "luonnollisiksi" **NP**-täydellisiksi ongelmiksi. Tällaisia ongelmia ovat muun muassa alussa mainitut verkkoteorian ongelmat.

Todetaan vielä, että **NP**-täydellisten ongelmien tapauksessa tarkistusrelaatiossa R esiintyvä etsimisongelma "annetulle x etsittävä y , jolle $R(x, y)$ " palautuu alkuperäiseen ongelmaan, eikä siis lisää vaativuutta. Erityisesti, jos $\mathbf{P} = \mathbf{NP}$, niin tälle etsimisongelmalle on jopa polynomiaikainen algoritmi.

Lienee vielä syytä huomauttaa, että on olemassa kieliä, joiden tiedetään olevan vaativuusluokassa **NP**, mutta joiden ei tiedetä olevan **NP**-täydellisiä tai luokassa **P**. Esimerkkinä tällaisesta kielestä mainittakoon yhdistettyjen lukujen joukko. Tässä tapauksessa tarkistusrelaatio R määräytyy kaavalla $R(\bar{a}, \bar{b}) \iff 1 < b < a$ ja $b|a$. (Tässä \bar{a} ja \bar{b} ovat lukujen a ja b koodeja.)

Vaikka tässä onkin puhuttu vain luokista **P** ja **NP**, on näiden vaativuusluokkien sekä ala- että yläpuolella (ehkä jopa välissäkin) koko joukko mielenkiintoisia vastaavia vaativuusluokkia, joihin liittyy täysin samanlaisia käsitteitä ja kysymyksiä. Miksi sitten juuri tämä kysymys on suuren huomion kohteena? Ensinnäkin $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ -kysymys lienee historiallisesti ensimmäinen alan (hyvin muotoilluista) kysymyksistä ja siksi suuren huomion kohteena. Toinen seikka liittyy vaativuusluokan **P** oleelliseen rooliin teoreettisessa tietojenkäsittelytieteessä: luokassa **P** laskettavat ongelmat ovat niitä, joita tietokoneella on järkevä ratkaista. Tosin tiedetään myös ongelmia, joille tunnettu polynomiaikainen algoritmi on varsin suuri, mutta eksponentiaalinen algoritmi varsin siedettävä käytännössä vastaan tulevilla tilanteilla. Tässä mielessä $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ -ongelmaa voidaan pitää myös

teoreettisena yksinkertaistuksena tärkeään vaativuusteoreettiseen kysymykseen, mitä voidaan tehokkaasti laskea.

Eräs varteenotettava viite vaativuusteoriaan on [Pap94]. Toinen mielenkiintoinen viite myös loogisesta näkökulmasta vaativuusteoreettisiin kysymyksiin on [Imm98].

C. Tulevaisuuden näkymiä

Paitsi, että $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ -ongelman ratkaisija saanee mainetta ja mammonaa, ei siinä vielä kaikki. Vaikkei kysymystä itsessään ole pystytty ratkaisemaan, ei viimeisen kolmenkymmenen vuoden aikana ihan laakereillakaan ole levätty. Sen sijaan on etsitty useita uusia ongelmia, joihin tällä kysymyksellä on yhtymäkohtia, sekä on tutkittu, mitä seurauksia ratkaisulla olisi, oli tämä ratkaisu sitten kumpi tahansa. (Kaikkihan muistavat, että Fermat'n suuri lausekin lopulta ratkaistiin, kun keksittiin sopiva paljon tutkittu matematiikan osa-alue, johon se liittyy.)

Yksi mahdollinen ja maininnanarvoinen lähestymistapa tähän kysymykseen ovat ns. *probabilistiset vaativuushuokat*, joiden suhdetta luokiin \mathbf{P} ja \mathbf{NP} on myös paljon tutkittu. Näistä ehkä mielenkiintoisin on luokka \mathbf{BPP} , joka on niiden kielten luokka, jotka voidaan tunnistaa satunnaisella polynomiaikaisella algoritmilla, joka kullakin syötteellä antaa korkeintaan eksponentiaalisen pienen virheen. (Esimerkiksi alkuluvut ovat luokassa \mathbf{BPP} , mutta niiden ei tiedetä olevan luokassa \mathbf{P} .) Toki on $\mathbf{P} \subseteq \mathbf{BPP}$ mutta tämän luokan tarkka suhde tarinamme pääluokkiin ei ole tiedossa.

Yleisesti toki uskotaan, että $\mathbf{P} \neq \mathbf{NP}$. Ja monet lohduttautuvat vielä sillä, että vaikka olisikin $\mathbf{P} = \mathbf{NP}$, \mathbf{NP} -täydellisten ongelmien polynomiaikaisuus luultavasti on niin suuri, ettei sillä käytännössä ole mitään merkitystä.

Mainitaan vielä lopuksi pari $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ -kysymykseen liittyvää seikkaa. Itse asiassa nykypäivämme jopa perustuu monelta osin olettamukseen $\mathbf{P} \neq \mathbf{NP}$. Tarkastellaanpa nimittäin päinvastaisen ratkaisun muutamaa seurausta. Lähes kaikki nykyisin tietoliikenteessä käytetyt sala- ja suojausmenetelmät (kuten RSA, DES) perustuvat vankasti olettamukseen $\mathbf{P} \neq \mathbf{NP}$. Jos päinvastainen tulos todistettaisiin, etenkin osa näistä menetelmistä joutaisi täysin romukoppaan. Tämä siis esimerkiksi hyllyttäisi nykyään kovin suosittua nettiasioimisen kokonaan.

Toinen positiivisen vastauksen seuraus pistäisi puolestaan (lähes kaikki) matemaatikot virattomiksi. Silloin nimittäin kaikki (nykyisin yleisesti hyväksytyn aksioomasysteemin ZFC:n) järkevän pituiset teoreemat voitaisiin jättää tietokoneen löydettäväksi, sillä nämä on helppo tarkistaa polynomiajassa. Erityisesti tämä koskee kaikkia tämän lehden artikkeleita. Kannattaa siis aloittaa näiden ongelmien ratkaiseminen $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ -kysymyksestä!

Kiitokset Alex Hellstenille, Taneli Huuskoselle ja Tero Tulenheimolle kommentista tämän artikkelin aiemmasta versiosta.

Viitteet

- [Coo71] S. Cook. The complexity of theorem-proving procedures. In *Conference Record of Third Annual ACM Symposium on Theory of Computing*, ss. 151–158, 1971.
- [GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco, 1979.
- [Imm98] N. Immerman. *Descriptive complexity*. Springer Verlag, Berlin, 1998.
- [Pap94] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- [Tur37] A. Turing. On computable numbers with an application to the entscheidungsproblem. In *Proceedings of the London Mathematical Society*, 2:42, ss. 230–265, 1936/37.