Algoritmit yläkoulussa: miksi ja miten opettaa niitä matematiikan tunneilla?

Veera Lupunen

Matemaattis-luonnontieteellinen tiedekunta, Helsingin yliopisto

Tiivistelmä: Tässä artikkelissa tarkastellaan algoritmin käsitettä sekä sitä, miten algoritmeja voidaan opettaa yläkoulussa. Algoritmit ovat osista koostuvia toiminta-ohjeita, joilla on tiettyjä ominaisuuksia kuten päättyvyys ja tuloksellisuus. Algoritminen ajattelu puolestaan on ajatusprosessi, jossa tarkoituksena on löytää ratkaisu johonkin laskennalliseen, algoritmin avulla ratkaistavaan ongelmaan. Algoritmit on mahdollista toteuttaa tietokoneella ohjelmoimalla. Perusopetuksen opetussuunnitelman perusteissa opetuksen tavoitteiksi määritellään toisaalta algoritmisen ajattelun taitojen kehittäminen ja toisaalta ohjelmointitaidon saavuttaminen. LUMATIKKA-kurssilla *Algoritmisen ajattelun kehittäminen* pyritään tukemaan opettajia näiden aiheiden opettamisessa sekä tutustutaan algoritmeihin ja saadaan konkreettisia ideoita niiden opetukseen. Algoritmeihin tutustumalla voi kehittää algoritmisen ajattelun taitoja, minkä lisäksi niiden tunteminen on osa yleissivistystä. Algoritmien opetus yläkoulussa tukeekin monien matematiikan opetuksen tavoitteiden saavuttamista.

Asiasanat: algoritmit, algoritminen ajattelu, ohjelmointi, yläkoulu

Yhteystiedot: veera.lupunen@helsinki.fi

1 Algoritmit ympärillämme

Algoritmeja on kaikkialla ympärillämme. Ne avaavat älypuhelimiemme lukituksen kasvojamme tarkastelemalla, ehdottavat suoratoistopalveluissa katsottavaa ja kuunneltavaa, arvioivat lainanmaksukykyämme sekä etsivät ystävän vinkkaaman kahvilan yhteystiedot netistä, vaikka kahvilan nimen oikeinkirjoitus olisikin sinnepäin. Ne myös ohjaavat meidät oikeaan paikkaan, kuljimmepa jalan, polkupyörällä tai bussilla. Vaikka erilaisia algoritmeja on kaikkialla, me emme tavallisesti näe niitä. Ne ohjaavat elämäämme ja valintojamme, mutta emme aina tiedä tarkalleen, miten. Algoritmeista on yhteiskunnassamme tullut elintärkeitä, sillä niiden avulla ohjataan niin tehtaita kuin maatalouttakin, niin tietoliikennejärjestelmiä kuin sähköntuotantoakin.

Tässä artikkelissa pohditaan algoritmeja osana perusopetusta. Tulisiko niitä opettaa ja jos kyllä, niin miten? Artikkelissa tutustutaan ensin algoritmin ja algoritmisen ajattelun käsitteisiin, annetaan konkreettisia esimerkkejä erilaisista algoritmeista sekä valotetaan algoritmien ja ohjelmoinnin suhdetta. Tämän jälkeen luodaan katsaus perusopetuksen opetussuunnitelman perusteisiin: mitä perusteissa sanotaan

algoritmeista, algoritmisesta ajattelusta ja ohjelmoinnista? Lisäksi kuvataan, miten LUMATIKKA-täydennyskoulutuksen kurssi *Algoritmisen ajattelun kehittäminen* on omalta osaltaan pyrkinyt vastaamaan edellä esitettyihin kysymyksiin ja miten se tukee opettajien valmiuksia näiden aiheiden opettamiseen. Lopuksi pohditaan algoritmien opettamista koulussa opetussuunnitelman perusteiden asettamissa raameissa ja annetaan pedagogisia ideoita algoritmien opettamiseen.

Algoritmien tuntemus on osa tämän päivän yleissivistystä, ja kaikille yhteisen perusopetuksen tehtävänä on tarjota elämässä tarvittavat tiedot oppivelvollisille. Artikkelissa algoritmeihin liittyvän opetuksen tarkastelu on rajattu yläkouluun (perusopetuksen vuosiluokat 7–9) keskittyen, vaikka kurssi *Algoritminen ajattelun kehittäminen* onkin suunnattu opettajille varhaiskasvatuksesta toiselle asteelle. Myös artikkelin esimerkit ja pedagogiset vinkit on pyritty valitsemaan yläkoulukontekstiin sopivaksi.

2 Algoritmit ja algoritminen ajattelu

Perusopetuksen opetussuunnitelman perusteet (2014) asettaa opetuksen yhdeksi tavoitteeksi oppilaiden algoritmisen ajattelun kehittämisen tukemisen (Opetushallitus, 2014, s. 375). Sanapari ”algoritminen ajattelu” koostuu kahdesta sanasta: ”algoritmi” ja ”ajattelu”. Seuraavassa tarkastellaan näistä monelle vieraampaa käsitettä, *algoritmia*. Tämän artikkelin näkökulma on nimenomaan algoritmeissa, joten jälkimmäiseen käsitteeseen, ajatteluun, luotava katsaus on lyhyempi.

2.1 Mitä ovat algoritmit?

Käsite *algoritmi* on monelle tuttu uutisotsikoista ja sitä käytetään arkipuheessa vaivatta: algoritmit tunnistavat kasvomme ja päättävät, mitä uutisia näemme. Kuitenkin, kuten usein tämänkaltaisten käsitteiden kohdalla on, algoritmin käsite pakenee määritelmiä, eikä tiedemaailmassa ole konsensusta sen täsmällisestä määritelmästä. Etymologiasta ei juuri ole apua: algoritmi-sanan kerrotaan juontuvan 700–800-luvuilla eläneen matemaatikon Muḥmmad ibn Mūsā al-Khwārizmīnin nimestä. Algoritmi-sanalla viitattiin hänen kehittämäänsä laskuoppiin (Knuth, 1973, s. 1).

Algoritmin käsitteestä voidaan kuitenkin sanoa ainakin seuraavaa: algoritmi on osista koostuva toimintaohje, joka toteuttamalla voidaan ratkaista jokin ongelma. Mainitut osat ovat täsmällisesti määriteltyjä, ne tulee toteuttaa tietyssä järjestyksessä ja niitä on äärellinen, etukäteen laskettavissa oleva määrä. Kaikkiin ongelmiin ei ole olemassa tai tiedossa ratkaisun tarjoavaa algoritmia, mutta kaikki algoritmit

ratkaisevat jonkin ongelman. Samaan ongelmaan voi olla olemassa myös useita algoritmeja. (Knuth, 1973, s. 4; Ukkonen, 2003, s.19; Sipser, 2006, s. 156–157; Cormen ym. 2009, s. 5; Laaksonen & Lupunen, 2021, s.72)

Algoritmi-käsite liitetään tyypillisesti ohjelmointiin, mutta algoritmeista kuulee puhuttavan myös varsin laveasti ja käsitteellä tarkoitettavan mitä tahansa toimintaohjetta, vaatteiden päälle pukemisesta huonekalujen kokoamisohjeisiin ja leivontaresepteihin. (vrt. esim. Sipser, 2006, s. 156; Knuth, 1973, s. 4) Lisäksi algoritmeja esiintyy myös matematiikassa, jossa puhutaan esimerkiksi *kertolaskualgoritmista* (Opetushallitus, 2014, s. 235) tai kahden luvun suurimman yhteisen tekijän löytävästä algoritmista (*Eukleideen algoritmi*) (Sipser, 2006, s. 156; Knuth 1973, s. 2–4).

Jotta ongelma voidaan ratkaista tietokoneen avulla, algoritmin jokainen askel tulee voida ratkaista laskemalla. Tietokoneella ratkaistavan ongelman tulee siis olla *laskennallinen* – sellainen, että sen voi palauttaa tiettyihin matemaattisiin laskutoimituksiin. (Cormen ym., 2009, s. 5) Tällaisia laskutoimituksia ovat esimerkiksi *yhteenlasku*, *kertolasku*, *kokonaisjako* sekä *vertailuoperaatiot* pienempi kuin (<), suurempi kuin (>) ja yhtä suuri kuin (=). (Cormen ym. 2009, s. 23–24) Laskutoimituksia yhdistämällä, toistamalla ja suorittamalla tietyssä järjestyksessä voidaan ratkaista hyvinkin monimutkaisia ongelmia.

Tietokoneelle tarkoitettuja algoritmeja voidaan luokitella käyttötarkoituksen mukaan. Samantyyppisiä algoritmeja voi olla useita, joista jokainen hieman eri käyttötarkoitukseen kehitetty tai edellistä tehokkaampi. Niiden avulla voidaan saada ratkaisuja ongelmiin, kuten ”Mikä on lyhin reitti kaupungista A kaupunkiin B, kun tunnetaan kaikki näiden kahden kaupungin välissä olevat kaupungit ja niiden väliset reitit (*reitinhakualgoritmit*)?”, ”Mihin ruutuun ristinolla-pelissä kannattaa tietyssä pelitilanteessa piirtää oma merkki, jotta voitto olisi taattu (*minimax-algoritmi*)?” tai ”Miten salaiseksi tarkoitettu viesti voidaan suojata niin, ettei kukaan muu kuin viestin tarkoitettu vastaanottaja pysty ymmärtämään sitä (*salausalgoritmit*)?”

2.2 Algoritmit – leivontareseptejä tietokoneelle

Tässä artikkelissa algoritmit ymmärretään erityisesti tietokoneilla ohjelmointiin tarkoitetuiksi toimintaohjeiksi. Algoritmien ymmärtämistä auttaa kuitenkin niiden vertaaminen muunlaisiin toimintaohjeisiin. Eräs hyvä verrokki on leivontaresepti, joka antaa tarkat ohjeet siihen, miten saadaan aikaiseksi herkullisia leivonnaisia. Leivontaresepti ratkaisee herkullisen leivonnaisen aikaansaamisen ongelman.

Leivontareseptejä tai muita arkipäivän toimintaohjeita ei kuitenkaan tavallisesti kutsuta algoritmeiksi. Tarkastellaan seuraavaksi algoritmin ominaisuuksia vielä hieman tarkemmin ja pohditaan, miten ja miltä osin leivontareseptiesimerkki sopii yhteen niiden kanssa. Tarkastelussa on hyödynnetty algoritmiikan suurmiehen, tietojenkäsittelytieteen emeritusprofessorin Donald Knuthin viisikohtaista kuvausta algoritmien ominaisuuksista. Myös Knuth pohtii reseptien ja algoritmien suhdetta. (Knuth, 1973, s. 4–9)

Ensinnäkin algoritmi on *päättyvä*. Sen jokainen osa toistetaan enintään äärellinen määrä kertoja, minkä jälkeen sen suoritus pysähtyy. Algoritmin suoritus ei siis kestä loputtomasti, vaan viimeistään hyvin pitkän ajan kuluttua se päättyy. (Knuth, 1973, s. 4–5) Päättyvyys pätee myös leivontareseptiin: vaikka olisi kyse kuinka monimutkaisesta ja aikaa vievästä leivonnaisesta tahansa, sen tekeminen päättyy lopulta ja leivonnainen on valmis.

Toiseksi algoritmin jokainen askel on tarkasti *määritelty*. Askeleiden suorittamisen tapa ei siis voi olla tulkinnanvaraista tai epäselvää, vaan ohjeen tulee olla riittävän yksityiskohtainen ja täsmällinen, että algoritmin suorittaminen johtaa aina samaan, oikeaan ratkaisuun. (Knuth, 1973, s. 5) Kotileipojille suunnatut reseptit saattavat hyvinkin olla varsin epätäsmällisiä ja vaatia ihmiseltä tulkintaa ja ennakkotietoja. Reseptissä voidaan esimerkiksi todeta, että kananmunat vaahdotetaan sokerin kanssa ja olettaa leipojan ymmärtävän, että munista käytetään vain pehmeä aines, eikä niitä yhdistetä sokerin kanssa kuorineen. Tässä suhteessa leivontareseptit tyypillisesti poikkeavat algoritmin määritelmästä.

Kolmanneksi algoritmille voidaan antaa ratkaistavaa tapausta kuvaava *syöte*, joka voi algoritmista riippuen olla esimerkiksi luku tai lukuja, sana tai sanoja tai lista lukuja. (Knuth, 1973, s. 5; Cormen ym., 2009, s. 5) Leivontaesimerkissä syötteen voi ajatella olevan raaka-aineet ja leivontavälineet, joita leivonnaisen aikaan saamiseksi tarvitaan: tyypillisesti näitä ovat esimerkiksi jauhot, kohotusaineet, kulhot ja vispilät.

Neljänneksi, kun algoritmin suoritus päättyy, saadaan *tulos*, joka kertoo ongelman ratkaisun kyseisessä tapauksessa eli annetulla syötteellä. Tulos voi syötteen tapaan olla algoritmista riippuen monenlaisia asioita, kokonaisluvuista kuviin. (Knuth, 1973, s. 5; Cormen ym., 2009, s. 5) Leivontareseptin päättyessä tuloksena on tyypillisesti leivonnainen – ainakin jos syötteenä on saatu kaikki tarvittavat aineet ja välineet, ja ohjetta on noudatettu.

Viidenneksi algoritmi on *tuloksellinen* eli sen ratkaisu on mahdollista saavuttaa. Knuthin käyttämä englannin kielen sana *effectiveness* on tässä hieman hankalasti

suomennettava, mutta jokin sarja ohjeita on algoritmi vain, jos sen jokaiseen osaan on olemassa algoritminen ratkaisu. Lisäksi jokaisen näistä osista tulee olla suoritettavissa äärellisessä ajassa. (Knuth, 1973, s. 6) Algoritmin osana ei siis voi olla esimerkiksi jokin ratkeamattomaksi osoitettu ongelma, kuten pysähtymisongelma: pysähtyykö minkä tahansa ohjelman suoritus aina kaikilla syötteillä (Sipser 2006, s. 175–185). Algoritmin osana ei voi myöskään olla esimerkiksi eettinen ongelma, kuten ”onko varastaminen oikein”, sillä eettisiä ongelmia ei voida ratkaista algoritmisesti.

Lisäksi on huomattava, että vaikka algoritmin toteuttaisikin tietokoneohjelmana, saman asian voi aina tehdä myös käsin, kynän ja paperin avulla. (Knuth, 1973, s. 6) Ratkaistavasta ongelmasta riippuen aikaa, kyniä ja paperia voisi kuitenkin kulua hyvin paljon, minkä vuoksi algoritmin toteuttaminen tietokoneella on tyypillisesti hyödyllistä. Paljon aikaa voi tarkoittaa tuhansia tai satoja tuhansia vuosia. Tehokkaalta tietokoneelta saman asian tekemiseen voi mennä huomattavasti vähemmän aikaa, esimerkiksi joitakin minuutteja. (ks. esim. Ukkonen, 2003, s. 20) Tietokoneet eivät myöskään tee huolimattomuusvirheitä, toisin kuin ihmiset. Oikean vastauksen saaminen tosin edellyttää sitä, ettei ihminen ole algoritmia tietokoneelle toteuttaessaan tehnyt ohjelmointivirheitä.

Leivontareseptin lisäksi toinen algoritmin käsitettä havainnollistava esimerkki on peräisin klassikkoteoksesta Linnunradan käsikirja liftareille. Kirjasarjan ensimmäisessä osassa supertietokone Syvä Miete saa tehtävän: sen tulee selvittää *Vastaus elämään, kaikkeuteen ja kaikkeen sellaiseen* (Adams, 1981, s. 161). Epäselväksi jää, mikä tässä on täsmällinen syöte. Vuosimiljoonien laskemisen jälkeen vastaukseksi saadaan tulos ”42”. Jos syöte olikin epämääräinen, tulos on sitäkin eksaktimpi, vaikkakin kontekstissaan käsittämätön. Tarina jatkuu niin, että tietokoneen seuraavaksi tehtäväksi annetaan selvittää se tarkka kysymys, johon 42 on vastaus – eli siis määriteltävä ratkaistava ongelma ja täsmällinen syöte. Kysymys ei kirjasarjan aikana ehdi selvitä.

2.3 Algoritminen ajattelu ja ongelmanratkaisu

Kuten edellä kerrottiin, algoritmi on osista koostuva ratkaisu johonkin ongelmaan. Ongelmanratkaisutaitojen kehittäminen on yksi matematiikan opetuksen tavoitteita perusopetuksen opetussuunnitelman perusteissa. (Opetushallitus, 2014, s. 17, 374) Sen sijaan perusteissa ei opetuksen tavoitteeksi aseteta esimerkiksi algoritmin käsitteen opettamista tai erilaisiin algoritmeihin tutustumista. Algoritmin käsitteen tunteminen sekä erilaiset algoritmit kuitenkin liittyvät algoritmisen ajattelun kehittämiseen (Yadav ym., 2017, s. 57).

Algoritminen ajattelu on perustavanlaatuinen ajattelun taito ja ajatusprosessi, johon liittyy kiinteästi ajatus ongelmien ratkaisemisesta. (Wing, 2006, s. 33; Selby & Woollard, 2013) Algoritmisessa ajattelussa ongelmia pyritään ratkaisemaan nimenomaan algoritmisesti. Ratkaistavien ongelmien tulee siis olla ratkaistavissa laskennallisesti, kuten todettua. Tämä ei kuitenkaan tarkoita sitä, että ongelmat tulisi ratkaista ohjelmoimalla, vaan laskennan voi hyvin tehdä myös ihminen. (Wing, 2006, s. 33)

Algoritmisen ajattelun taitoon kuuluu kyky ajatella abstraktisti sekä ongelman pilkkominen pieniin, ratkaistavissa oleviin osatekijöihin, jotka kootaan yhteen lopulliseksi ratkaisuksi. Tärkeää on kyky arvioida ratkaisua ja tarvittaessa korjata sitä. Algoritmiseen ajatteluun kuuluu myös olemassa olevien ratkaisujen yleistäminen: miten tiettyä tarkoitusta varten luotua ratkaisua voidaan käyttää muiden ongelmien ratkaisuun? (Selby & Woollard, 2013)

Algoritminen ajattelu liittyy ohjelmointiin ja algoritmeihin, mutta ei rajoitu niihin. Taitoa ratkaista ongelmia ja laatia vaiheittaisia ohjeita tarvitaan monessa muussakin yhteydessä. Se ei ole tietokoneen kaltaisesti ajattelua, vaan ongelmien ratkaisua luovasti, tehokasta ja riittävän tarkkaa ratkaisua etsien. Ratkaisussa voidaan käyttää sellaisia tietojenkäsittelytieteellisiä keinoja kuin toistaminen rekursiivisesti ja toimintojen rinnakkaistus. (Wing, 2006, s. 35)

Algoritmisen ajattelun harjoittelu on yksi keino kehittää ongelmanratkaisutaitoa (Yadav ym., 2017, s. 57–59). Algoritmin käsitteen tunteminen tarjoaa mallin siihen, miten ongelman voi ratkaista pilkkomalla sen pieniin osiin, joihin on olemassa ja tiedossa ratkaisu. Erilaisten algoritmien tunteminen taas tarjoaa esimerkkejä ratkaisutavoista, joita oppilas voi hyödyntää vastaan tulevien ongelmien ratkaisussa. Jos oppilas tuntee algoritmin, jonka avulla voidaan etsiä kaikki mahdolliset reitit kaupungista A kaupunkiin B, hän voi hyödyntää tätä algoritmia tilanteessa, jossa tulee etsiä lyhin mahdollinen reitti kaupunkien A ja B välillä. Näin ratkaistava ongelma voidaan palauttaa ongelmaan, johon tunnetaan jo ratkaisu.

3 Algoritmit ja ohjelmointi

Jotta tietokone voi ratkaista ongelman jonkin algoritmin mukaisesti, algoritmi pitää toteuttaa jollakin *ohjelmointikielellä*. Eräitä ohjelmointikieliä ovat Python, Java ja Scratch. Edellä esitetty fiktiivinen supertietokone Syvä Miete havainnollistaa asiaa. Syötteen (epämääräinen pyyntö Vastauksesta) ja tuloksen (42) väliin voi kuvitella ongelman ratkaisevan algoritmin. Syvä Miete on tietokone, ja sen ”sisällä” on *algoritmin mukaan tehty tietokoneohjelma*. Algoritmi itsessään on abstraktio, jonka perusteella

voidaan laatia algoritmin *toteuttava* tietokoneohjelma. Algoritmin voi kuvata esimerkiksi sanallisesti kertomalla vaihe vaiheelta, mitä ongelman ratkaisemiseksi tulee tehdä.

Algoritmit voidaan siis toteuttaa ohjelmointikielellä eli kirjoittaa jollakin ihmisten kehittämällä ohjelmointikielellä tietokoneen ymmärtämään muotoon. Perusopetuksen opetussuunnitelman perusteet asettavat algoritmisen ajattelun yhteydessä opetuksen tavoitteeksi myös *ohjelmointiin* liittyvät taidot (Opetushallitus, 2014, s. 375). Seuraavassa esitellään eräs algoritmi ja havainnollistetaan samalla sen ja ohjelmoinnin välistä suhdetta.

Määritellään ongelmaksi seuraava kysymys: ”Mitä tarkoittaa suomen kielen sana *vadelma*?” Oletetaan, että käytössämme on sanakirja, josta kyseinen sanan selitys löytyy. Tässä tilanteessa algoritmin syöte on siis sana vadelma ja tulos on sanaselitys, vaikkapa *mehukas ja hyvänmakuinen luumarja, tyypillisesti punainen*. Ongelma voidaan ratkaista käyttämällä jotakin hakualgoritmia, esimerkiksi *binäärihakua*. Nimen ei kannata antaa hämätä – se ei sinänsä liity mitenkään binäärilukuihin.

Binäärihakua voidaan käyttää, mikäli etsittävät asiat ovat jossakin sopivassa järjestyksessä toisiinsa nähden. Sanakirjassa sanat on yleensä järjestetty aakkosjärjestykseen. Binäärihaku on tehokas tapa etsiä jokin tietty tieto, joka on tallennettu johonkin *tietorakenteeseen*. Tietorakenne on tapa säilyttää tietoa; tässä se on sanakirja, joka koostuu peräkkäin liitetyistä sivuista, joilla on sivunumero. Sivut muodostavat aukeamia, joilla on sanoja selityksineen.

Algoritmin toiminnan voisi kuvata sanallisesti vaikkapa seuraavasti:

1. Avaa sanakirja keskikohdasta.
2. Jos etsitty sana on avatulla aukeamalla, lue sen selitys ja pysähdy.
3. Jos sana ei ole avatulla aukeamalla, tutki, onko sana aakkosissa ennen kyseisellä aukeamalla olevia sanoja vai niiden jälkeen. Ota se puoli sanakirjaa, jolla sana on. Tästä alkaen tutkitaan vain tätä osaa sanakirjasta. Unohda toinen puoli.
4. Toista kohtia 1.–3. kunnes löydät aukeaman, jolla etsimäsi sana on.

Tämän algoritmin sanallisen kuvauksen perusteella ihminen voi laatia algoritmista ohjelmatoteutuksen tietokoneelle. Pythonilla binäärihakualgoritmin toteutus voisi näyttää esimerkiksi seuraavalta:

```
sanakirja = [("kissa", "lemmikkinä pidetty pe-
toeläin"),("pulla", "makea leivonnainen"),("tuoli", "huo-
nekalu, jolla voi istua")]

sana = "kynä"

a = 0
b = len(sanakirja)-1

while a <= b:
    k = (a+b)//2
    if sanakirja[k][0] == sana:
        print(sanakirja[k][1])
        break

    if sana < sanakirja[k][0]:
        b = k-1
    else:
        a = k+1
```

Python-ohjelmaa voi halutessaan kokeilla itse joko omalla tietokoneella, jolle on asennettu Python, tai esimerkiksi LUMATIKKA-ohjelman ohjelmointikursseillakin käytössä olevan Tie koodariksi -sivuston editorissa.

Vähemmän tehokas tapa etsiä sana sanakirjasta olisi aloittaa kirjan alusta ja käydä sitä läpi sivu sivulta, kunnes oikea sana löytyy. Tämä veisi kuitenkin huomattavasti enemmän aikaa kuin etsiminen binäärihakua käyttämällä. Binäärihaku onkin tehokas tapa hakea tietoa silloin, kun se on järjestetty sopivalla tavalla, esimerkiksi aakkosjärjestykseen.

Työkseen ohjelmoivan eteen tulee melko harvoin tilanne, jossa algoritmin ohjelmatoteutus pitäisi toteuttaa itse alusta alkaen samaan tapaan kuin edellä esitetyssä esimerkissä on tehty. Algoritmeja sen sijaan tarvitaan ja käytetään jatkuvasti, mutta mikäli kyse on yleisestä algoritmista ja yleisesti käytetystä ohjelmointikielestä, joku toinen ohjelmoija on todennäköisesti jo tehnyt algoritmin *ohjelmatoteutuksen* kyseiselle kielelle ja laittanut sen muiden saataville. Tällöin puhutaan *kirjastokoodista* eli koodista, jonka joku toinen on tehnyt, ja jota voi käyttää oman koodinsa osana.

Edellisessä esimerkissä vaikkapa komento *print* (tulosta) on tällaista koodia. Komento kuuluu Pythonin peruskirjastoon ja sitä käyttämällä ei tarvitse määritellä

tulostusta itse alusta alkaen. Järjestämisalgoritmit ovat hyvä esimerkki algoritmeista, joita käytetään usein, mutta joista kannattaa harvoin tehdä oma ohjelmatoteutus itse. Eri algoritmien, algoritmityyppien sekä ominaisuuksien tunteminen on tarpeen, jotta ohjelmoija osaa valita ohjelmaansa parhaiten sopivan algoritmin. Toisaalta algoritmien kehityksessä ja tutkimuksessa ohjelmatoteutuksen osaaminen on hyvin olennaista.

4 Algoritmit perusopetuksen opetussuunnitelmassa

Kun perusopetuksen opetussuunnitelman uusia perusteita laadittiin 2010-luvun alussa, Suomesta puhuttiin monessa yhteydessä teknologian edelläkävijämaana; olivathan esimerkiksi puhelinyhtiö Nokia ja mobiilipeli Angry Birds saavuttaneet maailmalla hurjan tunnettuuden. Samaan aikaan tietokoneisiin liittyvän opetuksen kerrottiin olevan olematonta moneen muuhun, kuten naapurimaa Viroon, verrattuna (Malmberg, 16.10.2013). Uutisotsikot kertoivat niin koodaripulasta (Lappalainen, 13.4.2011) kuin siitäkin, että jo pienet lapset voisivat oppia ohjelmointia (Koistinen, 23.11.2013).

Vuoden 2014 alussa kerrottiin, että myös Suomen peruskouluissa aloitettaisiin uuden opetussuunnitelman myötä ohjelmoinnin opetus (Lehtinen, 21.1.2014). Perusteet otettiin käyttöön asteittain, ja tällä hetkellä ohjelmointia opetetaan kaikilla peruskoulun luokka-asteilla. Opetussuunnitelman perusteet asettavat opetukselle tavoitteet, sisällöt ja arvioinnin kriteerit. Seuraavaksi opetussuunnitelman perusteita tarkastellaan yläkoulun algoritmeihin ja ohjelmointiin liittyvän sisällön osalta.

4.1 Algoritmit, ohjelmointi ja tiedon käsittely perusopetuksen opetussuunnitelmassa

Perusopetuksen opetussuunnitelman perusteissa (2014) yläkoulun ohjelmointiin liittyvät tavoitteet ovat osa muiden oppiaineiden opetusta: ohjelmointia ”harjoitellaan osana eri oppiaineiden opintoja.” (Opetushallitus, 2014, s. 284) Opetuksen näkökulma on teknologioiden hyödyntämisessä ja käytännön taidoissa. Myös tietoturvaan ja vuorovaikutukseen liittyvät tavoitteet nostetaan esiin. Erillistä ohjelmoinnin tai tietotekniikan oppiainetta sen sijaan ei ole – enää. Tietotekniikan valinnaisaine oli vieraillut vuoden 1985 peruskoulun ja lukion opetussuunnitelmien perusteissa, mutta poistunut sieltä jo seuraavan opetussuunnitelmauudistuksen myötä vuonna 1994 (Laaksonen & Lupunen, 2021, s. 52–53).

Taulukossa 1 on kuvattu opetuksen tavoitteita ja sisältöä, oppimisen tavoitteita sekä arviointia algoritmisen ajattelun ja ohjelmoinnin osalta. Nämä kuuluvat erityisesti matematiikan sisältöalueeseen S1 *Ajattelun taidot ja menetelmät*. Yläkoulussa algoritmista ajattelua tulisi perusopetuksen opetussuunnitelman perusteiden mukaan syventää (Opetushallitus, 2014, s. 375), eli sitä pitäisi aiempien opintojen perusteella jo olla jonkin verran. Päättöarvioinnin kriteerien mukaan peruskoulunsa päättävän tulisi vähintäänkin "tunnistaa yksinkertaisen algoritmin askeleet" (Opetushallitus, 2020, 181).

Ohjelmointia harjoitellaan osana sisältöaluetta S1 *Ajattelun taidot ja menetelmät*. Tavoitteena on, että oppilas osaisi sekä ohjelmoida itse että käyttää valmiita ohjelmia ongelmien ratkaisussa ja matematiikan opiskelussa. Saadakseen päättöarvioinnissa arvosanan 5 oppilaan tulee vähintäänkin osata testata "ohjattuna valmiita ohjelmia" (ks. Taulukko 1) (Opetushallitus 2020, s. 181).

Oppimisen ja opetuksen tavoitteissa on siis opetussuunnitelman perusteiden näkökulmasta kaksi osaa: algoritmisen ajattelun periaatteet ja ohjelmointitaito. Päättöarvioinnin kriteereissä ohjelmointia ja algoritminen ajattelun osaamista ei kuitenkaan kytketä toisiinsa. Näin ollen kriteerien näkökulmasta toista voi osata osaamatta toista – oppilaalla voi siis olla algoritmisen ajattelun taitoja, vaikka hän ei osaisikaan ohjelmoida. Vastaavasti voi osata ohjelmoida, mutta algoritmisen ajattelun taidot voivat olla heikot tai olemattomat. Molempia tulee kuitenkin osata, eikä edes arvosanaa viisi saa ilman kumpaankin liittyvää osaamista. Perusteissa algoritmisen ajattelun taitojen harjoittelussa keskeistä on ongelmanratkaisutaito, ei suurien tietomäärien käsittely tai jonkin muun tietojenkäsittelytieteen osa-alueista. Samaan tapaan ohjelmointi liittyy perusteissa algoritmiseen ajatteluun, ei tiedon käsittelyyn.

Taulukko 1. Algoritmisen ajattelun ja ohjelmoinnin opetus perusopetuksen opetussuunnitelman perusteissa (Opetushallitus, 2014, s. 375) ja sen päättöarvioinnin kriteereissä (Opetushallitus, 2020, s. 181).

Opetuksen tavoite (T20)	Ohjata oppilasta kehittämään algoritmista ajatteluaan sekä taitojaan soveltaa matematiikkaa ja ohjelmointia ongelmien ratkaisemiseen
Opetuksen sisältö (S1 Ajattelun taidot ja menetelmät)	Syvennetään algoritmista ajattelua. Ohjelmoidaan ja harjoitellaan hyviä ohjelmointikäytäntöjä. Sovelletaan itse tehtyjä tai valmiita tietokoneohjelmia osana matematiikan opiskelua.
Oppimisen tavoite	Oppilas ymmärtää algoritmisen ajattelun periaatteita. Hän osaa lukea, kommentoida, tulkita, testata, suunnitella ja ohjelmoida pieniä ohjelmia, joilla ratkaistaan matemaattisia ongelmia.
Arvioinnin kohde	Algoritminen ajattelu ja ohjelmointitaidot
Päättöarvioinnin kriteerit, arvosana 5	Oppilas tunnistaa yksinkertaisen algoritmin askeleet ja testaa ohjattuna valmiita ohjelmia.
Päättöarvioinnin kriteerit, arvosana 7	Oppilas käyttää ehto- ja toistorakennetta ohjelmoinnissa sekä testaa ja tulkitsee ohjelmia.
Päättöarvioinnin kriteerit, arvosana 8	Oppilas soveltaa algoritmisen ajattelun periaatteita ja ohjelmoi pieniä ohjelmia.
Päättöarvioinnin kriteerit, arvosana 9	Oppilas hyödyntää ohjelmointia ongelmien ratkaisussa. Oppilas muokkaa ja kehittää ohjelmaa.

4.2 Ohjelmointi ja algoritmit tiedon käsittelynä – tietojenkäsittelytieteen näkökulma opetussuunnitelman perusteisiin

Tietojenkäsittelytieteen tieteenalan näkökulmasta erityisen kiinnostavaa opetussuunnitelman perusteissa on se, että algoritmista ajattelua tai ohjelmointia ei liitetä sisältöalueeseen S6 *Tietojen käsittely, tilastot ja todennäköisyys* (Opetushallitus, 2014, s. 375), vaikka nimen perusteella niin voisi helposti ajatella. Mitä ohjelmointi ja algoritmit oikeastaan ovat, jos eivät tietojen käsittelyä? Opetussuunnitelmassa tiedon käsittely näkyy ymmärrettävän kuitenkin ennen kaikkea käsin tehtävänä tiedon käsittelynä. Tietokoneen tai laskimen käyttöä ei toki kielletä, mutta ohjelmointia ei tässä yhteydessä mainita eikä sisältöaluetta S6 ja tavoitetta T20 liitetä toisiinsa.

Tietojenkäsittelytieteen näkökulmasta nämä kolme asiaa – tietojen käsittely, ohjelmointi ja algoritmit kuitenkin liittyvät kiinteästi toisiinsa. Tietojenkäsittelytiede on tiedekentällä suhteellisen uusi tulokas, mutta keskeistä siinä on tiedon käsittely. Esimerkiksi Helsingin, Itä-Suomen, Tampereen ja Turun yliopistot nostavat tiedon käsittelyyn liittyvät kysymykset esiin opintosuuntaa esitellessään: tieteenalaan liittyy muun muassa tiedon hallinta, tietoliikenne, tietoturva sekä tiedon esittäminen, optimointi, tallentaminen, järjestäminen ja muu käsittely. Samassa yhteydessä mainitaan

myös ohjelmointi, ohjelmistot ja algoritmit. (Itä-Suomen yliopisto, 9.4.2022; Helsingin yliopisto, 9.4.2022; Tampereen yliopisto, 9.4.2022; Turun yliopisto, 9.4.2022)

Tieteenalan näkökulmasta ohjelmointi ja algoritmit liittyvät siis luontevasti perusopetuksen opetussuunnitelman perusteiden sisältöalueen S6 sisältöihin: tiedon jäsentämiseen ja analysointiin, sen käsittelyyn, kuten keskiarvon tai mediaanin laskemiseen sekä diagrammien tuottamiseen. Myös muilla sisältöalueilla on monia kohtia, joihin ohjelmointi ja algoritmit voidaan liittää luontevalla tavalla; esimerkkinä mainittakoon jaollisuuteen, lukujonoihin ja alkutekijöihin jakamiseen liittyvät sisällöt.

Ohjelmoinnilla ja algoritmeilla on siis paljon annettavaa yläkoulun matematiikan opetukselle. Moni LUMATIKKA-kurssille osallistunut onkin kurssitöissään yhdistänyt ohjelmointia ja matematiikan sisältöjä toisiinsa kekseliäästi ja molempia oppisisältöjä hyödyttävällä tavalla. Osallistujat ovat kurssitöissään havainnollistaneet muun muassa geometriaa Pythonin *Turtle-kirjasto* apunaan, kerranneet koordinaatistoa Scratchillä ja syventäneet todennäköisyyksiä simulaatioiden avulla. Algoritmeihin liittyviä ja erityisesti yläkouluun soveltuvia pedagogisia ideoita esitellään tämän artikkelin lopussa.

5 LUMATIKKA-kurssi *Algoritmisen ajattelun kehittäminen*

Algoritmeihin liittyvä opettaminen voi tarkoittaa ainakin kahta asiaa: toisaalta se voi tarkoittaa yksittäisten algoritmien opettamista siten, että oppijalle esitellään algoritmin määritelmiä, erilaisia algoritmityyppejä ja algoritmeja sekä kerrotaan, mitä ongelmia ne ratkaisevat ja mihin niitä käytetään. Tällöin oppija saa käsityksen siitä, millaisia ratkaisuvaihtoehtoja on olemassa ja mahdollisesti myös, miten algoritmeista voidaan sanallisen kuvailun tai pseudokoodin avulla tehdä konkreettisia ohjelmatoteutuksia. Näin algoritmeja voidaan havainnollistaa ohjelmoinnin avulla. Esimerkiksi Helsingin yliopistossa tietojenkäsittelytieteen kandivaiheen opintoihin kuuluu algoritmeihin tutustuminen tästä näkökulmasta (Helsingin yliopisto, 29.4.2022).

Toisaalta taas voidaan opettaa algoritmista ajattelua ilman, että esitellään algoritmin määritelmää tai yhtään todellista algoritmia tai edes mainitaan ohjelmointia. Tällöin tavoitteena on ennemminkin oppia ajattelun taitoja, jotka ovat hyödynnettävissä monenlaisissa vastaantulevissa ongelmatilanteissa. Algoritmista ajattelua voidaan harjoittaa niin leikkien kuin pulmatehtäviäkin ratkaisemalla – ihan ilman tietokonettakin. Tämä lähestymistapa on nähtävissä perusopetuksen opetussuunnitelman perusteissa (2014).

Molempia näkökulmia voidaan perustellusti kritisoida ja puoltaa. Ilman konkreettisia esimerkkejä algoritmin käsite voi helposti jäädä hahmottomaksi ja algoritmien merkitys tavoittamatta. Mikäli esitellään vain yksittäisiä algoritmeja, voi osaaminen jäädä pinnalliseksi ja ongelmanratkaisutaidot vähäisiksi. Sekä todellisten algoritmien tuntemukselle että ongelmanratkaisutaitojen kehittämiselle on tarve nyky-yhteiskunnassamme. On tärkeää ymmärtää sekä sitä, miten algoritmit vaikuttavat elämäämme, että millaisia ongelmia niiden avulla voidaan ratkaista.

LUMATIKKA-kurssilla *Algoritmisen ajattelun kehittäminen* on lähdetty siitä ajatuksesta, että molemmat näkökulmat ovat tärkeitä eli sekä algoritmeihin tutustumisella että ohjelmoinnilla on paikkansa algoritmisen ajattelun kehittämisessä. Algoritmeihin tutustumalla kurssilainen saa alustavan käsityksen siitä, millaisia algoritmeja on olemassa ja millaisiin tarkoituksiin niitä voidaan käyttää. Ohjelmointia harjoittelemalla tulee harjoitelleeksi monia algoritmiseen ajatteluun liittyviä tärkeitä osatekijöitä. Näitä ovat muun muassa ongelman pilkkominen pieniin, ratkaistavissa oleviin osaongelmiin, abstraktin ajattelun taito sekä löydetyn ratkaisun yleistäminen niin, että sitä voidaan hyödyntää myös muissa yhteyksissä (Wing, 2006, s. 33; Selby & Woollard, 2013). Kurssin teemaosiot MOOC-oppimisympäristössä näkyvät kuvasta 1.

Kuva 1. Kurssin teemaosiot MOOC-oppimisympäristössä.

Kurssi aloitetaan ohjelmointiin tutustumisella. Aluksi pohditaan, miksi ohjelmointia oikeastaan kannattaa opiskella ja opettaa sekä tutustutaan valikoimaan ohjelmoinnin peruskäsitteitä. Peruskäsitteistä kurssilla esitellään *muuttujat*, *ehtolauseet*, *toistorakenteet* sekä *syöte* ja *tuloste*. Niin ikään käydään läpi joukko eri ohjelmointikielille tyypillisiä tietotyyppejä kuten *merkki*, *merkkijono*, *kokonaisluku*, *liukuluku*, *totuusarvo* sekä *lista*. Käsitteisiin tutustumisen jälkeen ohjelmointia

harjoitellaan MIT-yliopistossa opetustarkoitusta varten kehitetyllä *visuaalisella ohjelmointikielellä* Scratchilla.

Oman Scratch-ohjelman voi tehdä joko oman idean pohjalta itsenäisesti tai seuraamalla valmista ohjelmointiohjetta. Valmista ohjetta seuraamalla Scratchin käyttö tulee tutuksi varsin vaivattomasti, ja samalla edellä esitellyt ohjelmoinnin peruskäsitteet konkretisoituvat. Esimerkiksi tietojenkäsittelytieteen tiedeluokka Linkin laatima ohje keräilypelin suunnittelemiseen tutustuttaa ohjelmoijan muuttujiin, ehtolauseisiin ja toistorakenteisiin. Itsenäisesti oman suunnitelman pohjalta työskennellessä harjoituksessa korostuvat ongelmanratkaisutaitojen kehittäminen alkaen ohjelman rakenteen suunnittelusta ja vastaan tulevista ongelmatilanteista selviämisestä.

Monella kurssilaisista ei ole aiempaa kokemusta ohjelmoinnista, ja valmis ohje tarjoaa tärkeää tukea ohjelmoinnin aloitukseen. Ohjelmointi saatetaan kokea vaikeaksi aiheeksi, minkä vuoksi epäilyksenä voi olla, voiko sitä oppia itse lainkaan. Tehtävän tarkoituksena onkin ollut antaa onnistumisen kokemuksia, tukea osallistujien minäpystyvyyden tunnetta sekä vahvistaa ajatusta siitä, että jokainen voi oppia ohjelmoimaan, kunhan saa opetusta ja riittävästi tarvittavaa tukea.

Itsensä uuden opettelulle altistaminen voi tehdä hyvää myös oman opettajuuden kasvulle. Tämä on korostunut kurssilaisten vastauksissa erityisesti toisella LUMATIKKA-kurssilla, jolla tutustutaan ohjelmointiin: *Ohjelmoinnin merkitys matematiikan opetuksessa* -kurssilla moni osallistuja on kuvannut, miten Python-ohjelmointikielen tehtävien parissa tuli koettua monia sellaisia tunteita, joita on tottunut näkemään omissa oppilaissa. Tällaisia ovat olleet turhautumisen, epätoivon ja osaamattomuuden tunteet – tunteita, jotka lienevät tuttuja suurimmalle osalle ohjelmointia joskus kokeilleista. Moni on kertonut samaistuneensa kokemuksen jälkeen paremmin oppilaidensa tunteisiin sekä kertonut uskovansa kokemuksella olevan positiivinen vaikutus oppilaiden auttamiseen heistä vaikealta tuntuvien tehtävien kanssa.

Ohjelmoinnin perusteiden jälkeen Algoritmisen ajattelun kehittäminen -kurssilla tutustutaan algoritmeihin: Tehtävänä on tutustua tarkemmin yhteen algoritmiin ja kirjoittaa siitä lyhyt kuvaus muiden kurssilaisten ja kurssin ohjaajan luettavaksi. Tutustumalla valitsemaansa sekä muiden kurssilaisten valitsemiin algoritmeihin osallistuja saa kuvan siitä, millaisia algoritmeja on olemassa, miten nämä algoritmit suurin piirtein toimivat sekä millaisiin tarkoituksiin niitä käytetään.

Tehtävän haasteena on ollut osallistujalle mahdollisesti aivan uuteen asiaan tutustuminen, sen hahmottaminen, mitä algoritmi-käsitteellä ylipäätään tarkoitetaan ja mitkä kaikki ovat algoritmeja. Valinnan helpottamiseksi algoritmin on voinut valita

eri algoritmityyppejä edustavia algoritmeja sisältävältä listalta, joka on tarjonnut yhden näkökulman siihen, millaisia algoritmeja on olemassa. Tehtävän yhtenä tavoitteena on ollut myös algoritmeista kertomisen harjoittelu siten, että sellaisetkin kuulijat, joiden erityisosaaminen ei ole tietojenkäsittelytieteen alalla, voisivat ymmärtää, mistä on kyse. Tehtävän tuomia tietoja algoritmeista sekä algoritmeista kertomisen taitoa kurssilaiset voivat hyödyntää osana omaa opetustaan. Kun tuntee erilaisia algoritmeja ja algoritmityyppejä, voi antaa oppilailleen esimerkkejä jokaisen elämään vaikuttavista algoritmeista. Kurssilla onkin saatu lukea erittäin onnistuneita ja sopivan yksityiskohtaisia kuvauksia niistä.

Kurssin toisen kokonaisuuden muodostaa ohjelmoinnin opettaminen ja opetuksen suunnittelu. Kurssilaisia ohjataan pohtimaan sitä, minkä ikäisten kanssa ohjelmoinnin opetuksen voi aloittaa, mikä siinä on helppoa tai mahdollisesti vaikeaa sekä miten oppijoita voisi tukea oppimisprosessissa. Pohdinnassa voi myös reflektoida oppimistaan, onhan monella kurssilaisista vielä tuoreessa muistissa oma ensikosketus ohjelmointiin edellä kuvatun Scratch-tehtävän muodossa.

Ohjelmoinnin opetuksen suunnittelua pohjustetaan esittelemällä erilaisia ohjelmointikieliä ja *ohjelmointiympäristöjä*, joita koulun ohjelmoinnin opetuksessa voidaan hyödyntää. Tähän asti kurssilla opittua kootaan yhteen antamalla mielikuvituksen lentää: tehtävänä on pohtia, mitä sellaista haluaisi toteuttaa ohjelmoinnin, algoritmien ja teknologioiden avulla, joka ratkaisisi jonkin elämässä vastaan tulleen ongelman. Mielikuvitusta ruokkii myös tehtävä, jossa tutustuttavana on joukko eri lähteistä koottuja, ohjelmointiin ja algoritmeihin liittyviä leikkejä.

Kuvittelu- ja leikinvalintatehtävät pohjustavat harjoitusta, jossa tehtävänä on laatia tuntisuunnitelma ohjelmoinnin ja algoritmisen ajattelun kehittämiseen. Samoin kuin itse ohjelmoinnin, niin myös sen opettamisen on osa kurssilaisista kokenut vaativaksi epäillen omien taitojen riittävyyttä. Kurssilla kuitenkin halutaan rohkaista kokeilemaan, yrittämään, erehtymään ja oppimaan. Ohjelmoinnissa ja ongelmanratkaisussa erehtymistä tuskin voi välttää, eikä se aina tunnu mukavalta. Siksikin on tärkeää harjoitella myös epäonnistumista ja sitä, että siitä huolimatta yrittää uudestaan – näin myös opetuksessa. Kaikkea ei tarvitse osata, ei etenkään heti, eikä edes opettajan. Oppimista saa tapahtua niin oppilailla kuin opettajallakin opetuksen lomassa.

Tuntisuunnitelmatehtävässä kurssilaiset laativat kuvauksen oppitunnista tai oppituntikokonaisuudesta, jossa ohjelmointia opetetaan itsenäisenä aiheena tai osana jotakin muuta oppiainetta. Suunnitelmasta saadaan palautetta niin muilta

kurssilaisilta kuin kurssin ohjaajaltakin. Vuosien aikana luvan antaneiden kurssilaisten tuntisuunnitelmat on koottu yhteen kurssilaisten iloksi ja hyödyksi.

Kurssin viimeisissä osissa pohditaan eriyttämistä, luodaan katsaus tietoturvaan ja tietoliikenteeseen sekä pohditaan sitä, millaisessa ympäristössä itse opettaa algoritmeihin ja ohjelmointiin liittyviä sisältöjä. Kurssin yhtenä tavoitteena on saada ideoita oman opetuksen tueksi sekä rohkaistua jakamaan omia ajatuksiaan myös muille. Kurssin lopuksi osallistujia kannustetaankin toimimaan yhteistyössä ja jakamaan ideoita ja käytäntöjä niin omassa koulussa kuin muuallakin opettavien kollegojen kanssa. Tätä tavoitetta tukeakseen kurssin pohdintatehtävät palautetaan kaikkien kurssilaisten nähtävissä oleville keskustelualueille. Tehtävissä ääneen lausutut ideat tulevat siis jakoon kuin itsestään ja ovat sovellettavissa eri ikäisille oppijoille.

Kurssille osallistuneet ovat saadun palautteen perusteella kokeneet kurssin oman opetuksen kannalta hyödyllisenä: kurssi on innostanut aiheen parissa työskentelyyn, tarjonnut valmiita ideoita omassa opetuksessa hyödynnettäväksi ja vahvistanut luottamusta omaan osaamiseen. Toisaalta kurssin myötä on saattanut huomata, että opittavaa onkin vielä enemmän kuin oli ajatellut, ja että oma osaaminen on kurssin jälkeenkin vielä melko pintapuolista, eli ymmärtää oppimisen polun vasta alkavan. Parhaimmillaan kurssi on kuitenkin myös syventänyt ja selkeyttänyt algoritmeihin ja ohjelmointiin liittyvää ajattelua.

6 Pedagogisia ideoita – algoritmit opetuksessa

LUMATIKKA-kurssilla *Algoritmisen ajattelun kehittäminen* osallistujia haastetaan sitomaan ohjelmoinnin, algoritmien ja algoritmisen ajattelun opettaminen osaksi valitsemaansa oppiainetta – matematiikkaa tai mitä tahansa muuta ainetta. Alle kootut pedagogiset ideat ovat saaneet inspiraatiota pääosin kurssilaisten tehtävistä. Harjoitukset voi toteuttaa yksittäisinä oppitunteina tai esimerkiksi kiertopistetyöskentelynä. Pisteitä voidaan kiertää pareittain tai pienissä ryhmissä.

Salausalgoritmit

Salaperäiset viestit kiehtovat monia ja motivoivat oppimisen pariin. Viestien salausta ja salauksen purkamista voidaan harjoitella esimerkiksi yksinkertaisen ja oikeasti käytössä olleen *Caesar-salauksen* avulla. Siinä viesti salataan vaihtamalla kirjaimia toisiinsa tietyn siirtymän verran niin, että viimeiseen kirjaimeen päästäessä siirrytään seuraavaksi aakkosten alkuun. Jos siirtymä on 3, kirjain a vaihdetaan kirjaimeen d ja

kirjain b kirjaimeen e. Vastaavasti kirjaimesta ä tulee kirjain b. Salausta voidaan myös soveltaa niin, että siirtymä onkin taaksepäin, esimerkiksi -3. Salauksen purkaminen tapahtuu siirtymällä päinvastaiseen suuntaan.

Haastavampi, mutta tutustumisen arvoinen salausalgoritmi on alkulukuja ja lukujen yhteisiä tekijöitä hyödyntävä *RSA-salaus*[1]. Algoritmin käyttö voidaan tehdä kokonaan käsin tai siinä voidaan hyödyntää ohjelmointia esimerkiksi Pythonilla. Salauksessa luodaan kolmen luvun avulla kaksi salausavainta: julkinen ja yksityinen. Ideana on, että viesti salataan julkista salausavainta ja tiettyä laskukaavaa käyttäen ja puretaan yksityistä avainta sekä toista laskukaavaa käyttäen. Algoritmiin tutustumista varten opettaja voi luoda avaimet etukäteen, jolloin oppilaiden tehtäväksi jää laskukaavojen käyttäminen. Vaihtoehtoisesti salausta voidaan ensin harjoitella valmiilla avaimilla, minkä jälkeen jokaisen tulee muodostaa itse uusi avain.

Salausalgoritmeihin liittyvänä tehtävänä voi olla ensin salauksen purkaminen annetusta viestistä, minkä jälkeen oma (vastaus)viesti tulee salata samalla tavalla. Caesar-salauksen yksinkertaisuuden vuoksi sillä käsiteltävät viestit voivat olla pitkiäkin, kun taas RSA-salaukseen kannattaa valita suhteellisen pieniä, vaikkapa kuusinumeroisia lukuja. Kehykseksi voi halutessaan kehittää jännittävän tarinan, joka kertoo, miksi salausta välttämättä tarvitaan. Salausalgoritmeihin tutustuminen voidaan aloittaa Caesar-salauksella ja siirtyä sen jälkeen vaikeampaan RSA-salaukseen. Lopuksi voidaan pohtia tosielämän tilanteita, joissa salausta tarvitaan tai käytetään. Esimerkkejä voi etsiä niin vakoiluun kuin arkisempiinkin aiheisiin, kuten verkkopankkiin ja pikaviestimiin (esimerkiksi WhatsApp ja Telegram) liittyen.

Järjestämisalgoritmit

Algoritmeja voidaan opiskella myös toiminnallisesti. Erityisen hyvin tähän soveltuvat järjestämisalgoritmit, joissa esimerkiksi listan alkiot järjestetään suuruusjärjestykseen. Harjoituksessa oppilaat saavat kukin numerolapun ja asettuvat riviin satunnaiseen järjestykseen. Rivi kuvaa listaa ja oppilaat sen alkioita. Yksi oppilaista (tai oppilaspari) johtaa toimintaa seuraten annettua algoritmia vaihe vaiheelta ja kertoen, minne kenenkin pitää milloinkin liikkua. Käytettäviksi järjestämisalgoritmeiksi sopivat esimerkiksi *lomitus- tai pikajärjestäminen* tai yksinkertainen ja hauskasti nimetty mutta auttamattoman hidas *kuplajärjestäminen*.

[1] Kiitos Ari Virtaselle ideasta käyttää RSA-salausta kouluopetuksessa.

Järjestäminen voidaan toistaa useita kertoja, jolloin useammat oppilaista pääsevät tutkimaan algoritmia ja antamaan ohjeita. Mikäli aikaa on, mukaan voidaan ottaa algoritmien nopeuden vertailu, järjestämiseen kuluvaa aikaa mittaamalla tai siirtymisten lukumäärä laskemalla. Lisäksi voidaan tarkastella erikseen tilanteita, joissa oppilaat ovat lähes oikeassa järjestyksessä. Millä algoritmilla menee silloin eniten aikaa ja millä vähiten? Numerolappujen sijaan voidaan käyttää myös vaikkapa kirjaimia tai eri kokoisia esineitä – mitä vain, minkä voi laittaa yksiselitteiseen (suuruus)järjestykseen. Mikäli harjoitukset toteutetaan kiertopistetyöskentelynä, tällä pisteellä yhtä aikaa olevat ryhmät voivat yhdistyä pisteellä olon ajaksi.

Samaan tapaan voidaan tutustua myös binäärihakuun. Oppilailla olevat laput kuvaavat listalla olevia alkioita, joita voivat olla esimerkiksi numero, kirjain tai sana. Listaa kuvaavaan riviin asetutaan tällä kertaa järjestykseen ja alkiota kuvaavaa lappua pidetään taustapuoli ylöspäin. Algoritmia seuraava oppilas saa etsittäväkseen tietyn listalla olevan alkion. Kun päästään kohtaan, jossa tehdään tarkistus “onko listan tässä kohdassa oleva alkio sama kuin etsittävä”, lappua pitelevä oppilas kääntää lapun ja näyttää siinä olevan alkion. Mikäli harjoitukseen haluaa lisähaastetta ja syvyyttä, valittu algoritmi voidaan toteuttaa ohjelmoimalla esimerkiksi opettajan johdolla.

Järjestämisalgoritmeihin tai binäärihakuun tutustuminen tukee algoritmisen ajattelun kehittämistä. Ne tarjoavat esimerkin siitä, kuinka jokin ratkaistavana oleva ongelma (järjestykseen saattaminen) voidaan ratkaista pilkkomalla se pienempiin palasiin, osaongelmiin, ja yhdistämällä osaongelmien ratkaisu koko ongelman ratkaisuksi. Useampaan saman ongelman ratkaisevaan algoritmiin, esimerkiksi kahteen eri järjestämisalgoritmiin tutustuminen, taas havainnollistaa, että sama ongelma voidaan ratkaista useammalla eri tavalla. Ratkaisutapoja, algoritmeja, voidaan myös vertailla toisiinsa ja tutkia sitä, mikä niistä ratkaisee ongelman tehokkaimmin.

7 Lopuksi: Algoritmit ja matematiikan opetus

Algoritminen ajattelu ja ohjelmointi ovat osa erityisesti matematiikan oppiainetta. Käytännön opetusjärjestelyissä on koulujen välillä eroja. Opetussuunnitelman perusteiden sanamuodon ”osana eri oppiaineiden opintoja” voi ymmärtää ainakin kahdella tavalla: ohjelmoinnin harjoittelu voidaan toteuttaa erillisenä, erikseen arvioitavana kokonaisuutena tai sitoa opiskeltaviin aiheisiin niin, että molempia harjoitellaan yhtä aikaa.

Mikäli ohjelmoinnin opetus järjestetään erillisenä kokonaisuutena, oppitunneista osa varataan ohjelmoinnin harjoittelulle ja niillä keskitytään ohjelmointiin.

Luontevalta tuntuva malli on altis kritiikille: Opettajan näkökulmasta tämä tarkoittaa aiempaan opetussuunnitelmaan nähden vähemmän aikaa opetussuunnitelmassa mainituille muille sisällöille, sillä tuntijakoa ei muutettu tai matematiikan opetussisältöjä juuri karsittu opetussuunnitelmauudistuksen yhteydessä.

Oppilaan näkökulmasta taas ohjelmointi voi tuntua muusta oppisisällöstä irralliselta kokonaisuudelta, ja siihen liittyvät tiedot ja taidot voivat unohtua nopeasti opintojakson päätyttyä. Ohjelmoinnin ja algoritmisen ajattelun osaaminen tulee myös arvioida arviointikriteerien mukaisesti, eikä sitä siksi voi jättää ainakaan lukukauden aivan viimeisten oppituntien hauskaksi ja "ylimääräiseksi" aiheeksi. Toisaalta harkitusti toteutettuna ohjelmoinnin opetuskokonaisuus toimii hyvänä johdatuksena ohjelmointiin, tarjoaa mahdollisuuden oppia sen perusteet syvällisesti ja tuo esiin sen ominais- ja erityispiirteitä.

Vaihtoehtoisesti ohjelmoinnin opetus liitetään jatkuvasti osaksi muuta opetusta, eikä sitä ole erotettu omaksi kokonaisuudekseen. Tällöin ohjelmointi ja algoritmit näyttäytyvät parhaimmillaan hyödyllisinä ja hyödynnettävinä tietoina ja taitona, joita voi käyttää apuna aidoissa ongelmanratkaisutilanteissa. Ohjelmoinnin harjoittelu voi syventää esimerkiksi matematiikan sisältöjä ja tukea niiden oppimista. Toisaalta mikäli ohjelmoinnista ei missään kohtaa käydä systemaattisesti läpi perusteita, voi osaaminen jäädä toistamisen tasolle, eikä syvempää ymmärrystä ohjelmoinnin periaatteista synny. Oppilas voi esimerkiksi osata käyttää todennäköisyyttä simuloivaa ohjelmaa ilman, että ymmärtää, mitä mikäkin rivi oikeastaan tekee. Tämä voi riittää päättöarvioinnissa alimpien arvosanojen saamiseksi, mutta ylempiin vaaditaan jo syvempää osaamista – jota taas on vaikea saavuttaa ilman perusteiden hallintaa. Ohjelmointi voi myös näyttäytyä hahmottomana ja satunnaisesti toimivana asiana, jos sen kohtaa milloin minkäkin asian yhteydessä ilman, että yhteyttä on selväsanaisesti selitetty.

Parhaimmillaan algoritmien opetus tukee kuitenkin matematiikan opetuksen tavoitteiden saavuttamista – ei vain algoritmisen ajattelun kehittämisen, vaan monen muunkin sisältöalueen, alkuluvuista ongelmanratkaisutaitoihin. Algoritmeja siis kannattaa ja pitääkin opettaa yläkoulussa. Niihin voi tutustua ohjelmoinnin yhteydessä, mutta myös ilman tietokonetta. Erilaiset algoritmit ovat kiinteä osa arkeamme ja juhlaamme – siten on tärkeää tietää, mitä nämä algoritmit ovat ja miten ne toimivat.

Lähteet

Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. (2009). *Introduction to Algorithms* (3. painos). The MIT Press.

Helsingin yliopisto. (29.4.2022). *Tietojenkäsittelytieteen opintojen rakenne ja aikataulu*. https://www.helsinki.fi/fi/koulutusohjelmat/tietojenkasittelytieteen-kandiohjelma/opiskelu/tietojenkasittelytieteen-opintojen-rakenne-ja-aikataulu

Helsingin yliopisto. (9.4.2022). *Tietojenkäsittelytieteen esittely*. https://www.helsinki.fi/fi/matemaattis-luonnontieteellinen-tiedekunta/tiedekunta/tietojenkasittelytiede

Itä-Suomen yliopisto. (9.4.2022). *Tietojenkäsittelytieteen tutkinto-ohjelman opas*. https://www.uef.fi/fi/koulutus/tietojenkasittelytiede

Lappalainen, E. (13.4.2011). Pelialan kasvu imee osaajia. *Helsingin Sanomat*: Talous. https://www.hs.fi/talous/art-2000004800356.html

Koistinen, O. (23.11.2013). Jo 4-vuotias osaa ohjelmoida tietokonetta. *Helsingin Sanomat:* Teknologia. https://www.hs.fi/teknologia/art-2000002691046.html

Knuth, D. (1973). *The Art of Computer Programming*. Volume 1 / Fundamental Algorithms (2. painos). Addison-Wesley Publishing Company.

Laaksonen, A. & Lupunen, V. (2021). *Ohjelmointia oppimassa: Tietojenkäsittelytieteen tiedekasvatuksen vaiheita Helsingin yliopistossa*. Tiedeluokka Linkin 10-vuotisjuhlakirja. Helsingin yliopisto. http://hdl.handle.net/10138/337787

Lehtinen, T. (21.1.2014). Ministeri Kiuru: Ohjelmointi peruskoulun opetussuunnitelmaan. *Helsingin Sanomat*: Kotimaa. https://www.hs.fi/kotimaa/art-2000002704026.html

Malmberg, N. (16.10.2013). Suomen koulujen it-opetus Jordanian tasolla – uutta mallia haetaan Virosta. *YLE-uutiset*. https://yle.fi/uutiset/3-6885099

Opetushallitus. (2020). Perusopetuksen päättöarvioinnin kriteerit. *Opetushallituksen määräys 2020:5042*. https://www.oph.fi/sites/default/files/documents/Perusopetuksen%20päättöarvioinnin%20kriteerit%2031.12.2020_0.pdf

Opetushallitus. (2014). Perusopetuksen opetussuunnitelman perusteet. *Määräykset ja ohjeet 2014:96*. https://www.oph.fi/sites/default/files/documents/perusopetuksen_opetussuunnitelman_perusteet_2014.pdf

Selby, C. & Woollard, J. (2013). *Computational thinking: the developing definition University of Southampton*. https://eprints.soton.ac.uk/356481

Sipser, M. (2006). *Introduction to the Theory of Computation* (2. painos). Course Technology.

Tampereen yliopisto. (9.4.2022). *Tietojenkäsittelytieteiden koulutuksen esittely*. https://www.tuni.fi/fi/tule-opiskelemaan/tietojenkasittelytieteiden-koulutus

Turun yliopisto. (9.4.2022). *Tietojenkäsittelytieteen opiskelun esittely*. https://www.utu.fi/fi/opiskelijaksi/tietojenkasittelytiede-luonnontieteiden-kandidaatti-ja-filosofian-maisteri-3-v-2-v

Ukkonen, E. (2003). Mihin algoritmeja tarvitaan*? Tieteessä tapahtuu, 21*(7), 19–22. https://journal.fi/tt/article/download/57286/19319, viitattu 31.3.2022

Yadav, A., Stephenson, C., Hong, H. (2017). Computational thinking for teacher education. Commun. *ACM, 60*(4), 55–62. https://doi.org/10.1145/2994591