TARTU ÜLIKOOL

Arvutiteaduse instituut

Informaatika õppekava

Carola Kesküla

Rekursiooni käsitlemine selle õpetamisel

Bakalaureusetöö (9 EAP)

Juhendaja: Ahti Peder

Tartu 2019

Rekursiooni käsitlemine selle õpetamisel

Lühikokkuvõte:

Rekursiooni omandamist käsitlev (eestikeelne) algoritmidealane kirjandus on hõre. Käesolev töö uurib, kuidas Tartu Ülikool läheneb rekursiooni õpetamisele ja võrdleb seda Eesti ning välisülikoolide käsitlusega. Uurimus analüüsib, millised ülesanded rekursiooni tarvilikkusest märku annavad, milliseid tehnilisi oskusi tudengitega esitatud ülesannetega treenitakse ning milliste meetoditega õpetatakse tudengeid iseseisvalt rekursiooni rakendama. Töö eesmärk on parandada Tartu Ülikooli informaatikute valmisolekut kursuseks „Algoritmid ja andmestruktuurid“. Eesmärk saavutatakse pakkudes töö viimases peatükis välja komplekt ülesandeid, mis õppuritele vajaminevaid oskusi treenib.

Võtmesõnad:

Rekursioon, rekursiooni õpetamine, rekursiooni käsitlemine, rekursiivsed probleemid, rekursiivsed ülesanded, algoritmid, algoritmid ja andmestruktuurid

CERCS: P175 Informaatika, süsteemiteooria

Approaches in Teaching Recursion

Abstract:

The literature on algorithmic teaching of recursion (in Estonian) is minimal. This paper examines how University of Tartu approaches teaching recursion and compares it to the ways of Estonian and foreign universities. The study analyses which exercises indicate the necessity of using recursion, which technical abilities those exercises train and which tutoring methods are used to teach students the ability of applying recursion independently. The aim of the thesis is to improve University of Tartu's students' preparation for the course "Algorithms and Data Structures". The purpose is achieved in the last chapter of the paper by providing a set of exercises which trains pertinent skills.

Keywords:

Recursion, teaching recursion, recursive problems, recursive exercises, algorithms, algorithms and data structures

CERCS: P175 Informatics, systems theory

Sisukord

1. Sissejuhatus

Rekursiooni omandamist käsitlev algoritmidealane kirjandus on hõre. Seetõttu osutub metoodilise käsitlusviisi ülesehitamine suureks proovikiviks. Praegu õpivad Tartu Ülikooli informaatikud rekursiooni põhjalikult uurimata struktuuri järgi. Hüpoteesi kohaselt on tarvis rekursioonile pöörata rohkem tähelepanu kuna ainetevahelise koordineerituse puudumise tõttu ei jõua tudengid tasemele, kus oskaksid tehnikat ka iseseisvalt rakendada.

Käesolev informaatikateadmistele keskenduv uurimustöö uurib ja võrdleb Tartu Ülikooli käsitlust Eesti ja välisülikoolide käsitlusega. Töö fookus on õpetamisel esitatud ülesannetel ja tehnilistel oskustel, millele läbi ülesannete pööratakse tähelepanu. Töö käigus vastatakse kolmele uurimusküsimusele:

1. Milliseid rekursiooniga seonduvaid ülesandeid ülikoolid esitavad ja milliseid tehnilisi oskusi nendega oluliseks peetakse?
2. Millised esitatud ülesanded annavad märku, et probleem nõuab rekursiivset lahendusmeetodit?
3. Milliste meetodite ja materjalidega õpetatakse tudengeid iseseisvalt aru saama, et probleem nõuab rekursiooni?

Uurimustöö eesmärk on parandada Tartu Ülikooli tudengite valmisolekut kursuseks „Algoritmid ja andmestruktuurid“. Eesmärk saavutatakse pakkudes töö viimases peatükis uuritu põhjal välja komplekt ülesandeid, mis õppuritele vajaminevaid oskusi treenib.

Tasub mainida, et uurimus ei ürita väita seda, et rekursioon on igas olukorras otstarbekam kui iteratiivne programmeerimine. Töö heidab valgust olukordadele, millal rekursioon on ratsionaalsem valik ning sellele, kuidas taoliste olukordade äratundmist paremini õpetada. Vähendades rekursioonialast teadmatust ja struktureerides õpet veelgi saab Tartu Ülikool pakkuda oma tudengitele võimalikult kvaliteetset haridust.

2. Mõisted ja terminid

Rekursioon (ingl *recursion*) on tehnika, mille raames programmi funktsioon, protseduur või meetod kutsub iseennast välja [1]. Rekursiivne funktsioon peab täitma järgnevat kolme tingimust, et protsess lõputult ei korduks [2]:

1. Funktsioonis peab leiduma vähemalt üks baastingimus, mille kehtivusel funktsioon lõpetab ja milleni jõudmine on võimalik rekursiooni kasutamata.
2. Funktsioon peab põhinema suuresti mingil rekurrentsel seosel.
3. Järjestikused rekursiivsed funktsioonikutsed peavad arenema baastingimuse suunas.

Rekursioonist võib mõelda ka kui jaga-ja-valitse (ingl *divide-conquer-glue*) tehnika erijuhust, kus käesolev probleem jagatakse väikesemateks probleemideks, mida sõltumatult lahendatakse ja mille lahendused hiljem algse probleemi lahenduse leidmiseks „kokku liimitakse" [3].

Baasjuht (ingl *base case*), ka **baas** – rekursiivse funktsioonikutseta stsenaarium rekursiivses algoritmis või programmis, mis põhjustab funktsioonikutsete ahela lõppemist [2, 22].

Rekurrentne seos (ingl *recurrence relation*) – võrrand, mis määrab jada reegli põhjal, kuidas esitub järgmine termin eelmis(t)e termini(te) funktsioonina [95].

Aktiveerimiskirje (ingl *activation record*) – „tegumi või alamprogrammi eksemplari asendav andmeobjekt, mis sisaldab selle eksemplari andmeväärtusi ja protsessi oleku-andmeid. Aktiveerimiskirje võib sisaldada parameetreid, tulemeid, lokaalandmeid jm.[1]"

Rekursioonipuu – „konstrueeritakse mingi rekursiivse funktsiooni *f* (või rekursiivse protseduuri) jaoks selgitamaks, mis järjekorras ja milliste sisendparameetri(te) väärtus(t)ega toimuvad funktsiooni (protseduuri) rekursiivsed väljakutsed selle konkreetse rakenduse, st konkreetsete sisendandmete korral" [18:25].

Madal rekursioon (ingl *flat recursion*) – sisendandmestruktuuri elementide käsitlemine aatomitena (st. lihttüübi elementidena nt. sümbolitena) [96].

[1] IT terministandardi sõnastik. http://www.eki.ee/dict/its/index.cgi?Q=activation+record&F=M&C06=et&C10=1.

Sügav rekursioon (ingl *deep recursion*) – terve sisendandmestruktuuri käsitlemine võttes arvesse, et iga element võib olla omakorda andmestruktuur [96].

Rekursiivne generaator (ehk rekursiivne generaator-klass) – „on klass, milles leidub (vähemalt) isendimeetod *next*(). Konstruktorile antakse andmed, mis iseloomustavad mingit abstraktset, veel mitte eksisteerivat andmekogumit. [97]"

Listi tasandamine (ingl *list flattening*) – *n*-tasemelise listi ühetasemeliseks muutmine.

Põimemeetod (ingl *merge sort*) – järjestamismeetod, mille raames jagatakse elemendid võimalikult võrdselt kahte listi, mis omakorda samal viisil järjestatakse ning põimitakse.

Kiirsorteerimine (ingl *quicksort*) – järjestamismeetod, mille raames jaotatakse elemendid mingi võtme *x* järgi kolme rühma nii, et esimeses on kõik *x*-ist väiksemad, teises võrdsed ja kolmandas suuremad. Seejärel rakendatakse sama meetodit mõlemale äärmisele rühmale eraldi ja tulemused konkateneeritakse.

Lihtahel (ingl linked list) – ahel, milles „tipud ei kordu, v.a alg- ja lõpptipp, mis tohivad omavahel kokku langeda [66]".

Täielik otsing (ingl *exhaustive search)* - ülesanne, kus samm sammu haaval võetakse arvesse kõiki võimalusi lahenduseni jõudmiseks [63].

Väljakutsekaader (ingl *call frame)* – osa mälust, mida kasutatakse, et jälgida käesolevat funktsioonikutset. Väljakutsekaader "sünnib" funktsiooni väljakutsega ning "sureb" selle tagastusega. Iga väljakutsekaader sisaldab funktsiooni nime ja reanumbrit koodis, kus peale tagastust jätkata [100].

Memoiseerimine (ingl *memoization)* – programmeerimistehnika, mille raames salvestatakse tehtud arvutused vahemällu, et neid saaks vajadusel kiiresti uuesti arvutamata välja otsida [95].

3. Taustainfo

Eestikeelne kirjandus rekursiooni käsitlemisest on minimaalne. Antud peatükk selgitab, miks rekursiooni õpetamine on tarvilik, kuid keerukas ja miks tasuks ka Tartu Ülikoolis (edaspidi TÜ) tehnikale rohkem rõhku panna. Lisaks loob peatükk aluse edasisele tööle pakkudes ülevaate erinevatest viisidest, kuidas rekursiivseid ülesandeid klassifitseerida ning tuues välja tarvilikud kitsendused ja töö uurimuses kasutatud andmekogumise metoodika.

3.1 Miks on rekursiooni käsitluse uurimine oluline?

Rekursiooni olulisus

Rekursioon ei ole tänapäeval enam üksnes oluline matemaatiline mõiste vaid on ka populaarne programmeerimistehnika, tööriist hädavajalike probleemide lahendamiseks ja viis algoritmiliselt mõelda [3].

TÜ tudengid soovitasid aine LTAT.03.001 „Programmeerimine“ tagasisides kaas-tudengitel rekursioonile „eksamiks õppides rõhku panna ja üritada sellest lõplikult aru saada” [20] ning rekursiooni mitte alahinnata, sest „see on keeruline, aga oluline teema” [20]. Aine LTAT.03.005 „Algoritmid ja andmestruktuurid“ (edaspidi AA) tagasisides andsid tudengid aga nõu, et kui „rekursioon selgeks saada, siis läheb ka ainega kenasti” [21] ja, et „kindlasti ilma rekursiooni õppimata seda ainet ei läbi!” [21] Tudengid mõistavad, et rekursioon on keerukas, ent vajalik osa nende õpiteekonnast.

Suur hulk probleeme, millele lähenetakse rekursiivselt, leiaksid lahenduse ka iteratiivset programmeerimistehnikat (st. tsükleid) kasutades. Teisalt leidub rohkelt ülesandeid, mida kas tsüklitega ei saa väiksemateks tükkidest jagada või mille tsüklipõhine lahendus on tarbetult keerukas [4]. Näide sellisest probleemist on failipuu läbimine ja sealt faili otsimine. Konkreetsed probleemid, mis nõuavad rekursiooni kasutamist probleemi sisemise rekursiivse loomuse tõttu on näiteks fraktaalid, Hanoi tornide ülesanne ning failikataloogi suuruse arvutamine [10]. Kuna leidub fraktaale ja failipuid kasutavaid suuremahulisi rakendusi, siis ei ole rekursiooni kasutamine alati ei valikuline ega triviaalne.

Niisama oluline on tõsiasi, et enamus levinud fundamentaalseid programmeerimisstiile nagu imperatiivne, funktsionaalne (edaspidi FP) ja objektorienteeritud programm-

meerimine (edaspidi OOP) rakendavad rekursiooni. Ennekõike on sagedase rekursiooni kasutuse tõttu tuntud FP [11].

Mitmed autorid usuvad, et rekursioon võimaldab rafineeritud selgitusi keerulistele algoritmidele ja andmestruktuuridele [3, 5]. Probleemi rekursiivne lahendus on vähemalt sama elegantne või isegi elegantsem, kui kõik teised lahendusviisid. Ühtlasi on rekursioon põnev tehnika, mida uurida ja millega enda mõtlemisoskust arendada [5].

Rekursiooni õpetamise keerukus

Middlesexi Ülikooli arvutiteadlase ja kaasprofessori doktor G. George'i [12] arvates usutakse, et üleminekut rekursiooni mõiste arusaamiselt rekursiivse programmeerimise õppimisele peaks käsitlema ettevaatlikult. Ta usub, et rekursioonist arusaamine tekitab värsketele arvutiteaduse õppuritele kognitiivseid raskusi [6]. Nimetatud raskusi põhjustavad vähemalt kolm järgmist faktorit [5]:

1. lünk lihtsa rekursiivse algoritmi ja keeruka rekursiivse algoritmi teostusprotsessi vahel: Rekursiooni mõistmiseks peab õppur oskama eristada programmi (või meetodi) väljatrükki selle rekursiivsest protsessist (st. algoritmi täidesaatmisest), mis mõlemad nõuavad eriviisi mõistmist olenevalt õppijate kognitiivsetest mudelitest ja võimetest;
2. õppijate vigased või puudulikud arusaamad rekursiivse programmi täidesaatmise kohta;
3. hõre rekursiooniteemaline pedagoogika.

Seega on rekursiooni õppimine ja õpetamine muu hulgas võimas tööriist iseenda kognitiivsete oskuste edendamiseks. Viimane punkt illustreerib miks, arvestades ka puudulikku emakeelset kirjandust ja uurimust, on rekursiooni käsitlemise uurimine oluline, et TÜ suudaks oma tudengitele pakkuda võimalikult tulemuslikku arvutiteaduse haridust.

3.2 Rekursiooni klassifitseerimine

Rekursiivseid algoritme saab rühmitada mitmeti. Rühmitamise eesmärk on nii lugeja silmaringi avardamine kui ka arusaam klassifitseerides, kuidas mitmesugused rekursiivsed ülesanded ja algoritmid aitavad tudengitel keskenduda probleemi tükeldamisele, sügavuti mõistmisele ja funktsionaalsele abstraktsusele.

Rubio jt. [3, 7] väljapakutud rekursiooni klassifikatsioon on:

4. *Lineaarne rekursioon* või *unaarne rekursioon* – toimub ainult üks rekursiivne funktsioonikutse. Lineaarse rekursiooni alamtüüp on sabarekursioon (ingl *tail recursion*), milles rekursiivne funktsioonikutse on funktsiooni viimaseks käsuks. Populaarseks näiteks on faktoriaalide arvutamine.
5. *Mitmekordne* või *eksponentsiaalne rekursioon (sh. tenaarne rekursioon)* – toimub üle ühe rekursiivse funktsioonikutse. Näide mitmekordsest rekursioonist on hulga *n* elementide kõikide permutatsioonide leidmine.
6. *Mitmeetapiline* (ingl *nested*) *rekursioon* – rekursiivne funktsioonikutse on üks funktsiooni argumentidest. Ackermanni funktsioon (1) on näide mitmeetapilisest rekursioonist. Mitmeetapilist rekursiooni kasutatakse reeglina harva [7].

$$A(m,n) = \begin{cases} n+1, & \text{kui } m = 0; \\ A(m-1,1), & \text{kui } m > 0 \; ja \; n = 0; \\ A\big(m-1, A(m,n-1)\big), & \text{kui } m > 0 \text{ ja } n > 0. \end{cases} \quad (1)$$

Joonis 1. Ackermanni funktsioon [3].

7. *Mitmepoolne* (ingl *mutual*) *rekursioon* – antud funktsioon kutsub iseenda asemel teist funktsiooni välja, mis omakorda kutsub välja esimest funktsiooni. Mitmepoolne rekursioon võib koosneda enam kui kahest funktsioonist. Kasutusel tihti edasijõudnutele mõeldud arvutiteaduse teemades (nt. parserid[2]) [3, 7]. Lihtsamat sorti näide sellisest rekursioonist on naturaalarvu paarsuskontroll (2):

$$on\_paaritu(n) = \begin{cases} FALSE, & \text{kui } n = 0; \\ on\_paaris(n-1), & \text{kui } n > 0. \end{cases} \quad (2)$$

$$on\_paaris(n) = \begin{cases} TRUE, & kui\ n = 0 \\ on\_paaritu(n-1), & kui\ n > 0 \end{cases}$$

Joonis 2. Mitmepoolne rekursioon naturaalarvu paarsuskontrolli näitel [7].

Viimast kahte viisi peetakse eriliselt keerukaks ja seetõttu üldiselt arvutiteaduse madalama- ja kesktaseme kursustes ning õpikutes üldiselt ei käsitleta [7].

[2] „Programmeerimises nimetatakse parseriks tarkvarakomponenti stringide," nt. süntaksielementide, „otsimist andmetest." http://www.vallaste.ee/.

Teise viisi rekursiivseid programme eristada pakub välja Morazán [3, 8]:

- *Strukturaalne rekursioon* – argumendid, mida rekursiivse funktsiooni funktsioonikutsel edastatakse, on kas muutumatud või rekursiooni baasjuhule lähenevad. Kui rekursiivse funktsiooni parameetriks peaks olema näiteks andmestruktuur, siis rekursiivne funktsioonikutse toimub üle antud struktuuri pärisalamrühma (ingl *real subgroup*).
- *Generatiivne rekursioon* – argumendid, mida rekursiivsel funktsioonikutsel edastatakse, arvutatakse igal funktsioonikutsel uuesti ja seega ei tagata, et parameetrite muutus lõpeb rekursiooni baasjuhus. Näide sellisest rekursiooni kasutamisest on Collatzi hüpotees[3] ja Eukleidese algoritm arvutamaks positiivsete täisarvude suurimat ühistegurit.
- *Akumulatiivne rekursioon* – argumentide seas, mida rekursiivse funktsioonikutsel edastatakse, on üks või enam lisaparameetrit, mis sisaldab endast probleemi osalist lahendust.

Peale mainitute on oluliseks rekursiooniskeemiks puurekursioon ehk hargnev rekursioon. Tegu on rekursiivse funktsiooniga, milles võib olla mitu rekursiivset funktsioonikutset. Skeemi nimetus tuleneb sellest, et „graafilises esituses moodustub funktsiooni funktsioonikutseid tähistavatest nooltest puutaoline kujutis" [13]. Näide puurekursioonist on funktsioon, mis arvutab Fibonacci arvjada *n*-nda liikme.

3.3 Kitsendused

Käesolev töö uurib ja analüüsib ülikoolides esitatud rekursiooniga seonduvaid ülesandeid võttes arvesse järgnevaid kitsendusi ja põhimõtteid:

1. Selguse ning struktuuri eesmärgil on arvesse võetud ainult valimülikoolide kohustuslikud õppeaineid. Sama printsiip rakendub ülesannetele, õppematerjalidele ja –viisidele st. antud lõputöö ei võta arvesse lisaülesandeid ja soovitatavaid materjale. Kitsenduse eesmärk on tööga pakkuda kohustuslikult

[3] Hüpotees, mille kohaselt on „antud on naturaalarv n. Kui n on paarisarv, siis jagatakse ta 2-ga. Kui n on paaritu arv, siis asendatakse ta arvuga 3n+1. Seni tõestamata hüpoteesi kohaselt lõpeb see protsess ükskõik millisest arvust lähtudes alati arvuga 1". http://kodu.ut.ee/~eno/progral13/Progralused2013loeng4.pdf.

läbitavate õppekavade põhjal võimalikult objektiivne ja struktureeritud ülevaade uurimisküsimustest.

2. Ülesandeid klassifitseeritakse vastavalt ülesande kirjeldusele st. olenemata töö autori aimdusest tehnilise oskuste osas peab ülesande püstitus oskusele vihjama. Kitsenduse eesmärk on vältida eeldusi, et materjaliga töötav tudeng on samuti antud tehnilisest oskusest teadlik.
3. Kui ülesande eesmärk ei olnud defineeritud, siis klassifitseeriti see tehnilise oskuse „baasjuhu ja sammu äratundmine ning mõistmine" alla kuna nimetatud oskus on rekursiooni elementaarseim teadmine.
4. Töö põhineb vabalt kätte saadud materjalil. Töösse kaasatud ülikoole on eelistatud sellel põhimõttel, et neil oleks võimalikult palju vabalt kättesaadavat materjali.
5. Töös on kasutusel hiliseim kättesaadav materjal (nt. kui õppeaine koduleheküIge 2018. aastast ei ole, aga on saadaval lehekülg aastast 2016, siis on kasutusel viimane).

3.4 Uurimuse mall

Töös rakendatud uurimuse mall kasvas välja mitmekordse andmekogumise ja -sünteesi tulemina. Mall põhineb esmauurimuse käigus selgunud (töö autori hinnangul) olulistel tehnilistel oskustel.

Iga valimülikooli igat õppeainet on uuritud võimalikult põhjalikult järgneva malli alusel:

- Õpiväljundid ja aine sisu kirjeldus;
- Õppeviisid;
- Õppematerjalid;
- Esitatud ülesanded;
    - Kõik esitatud ülesanded;
    - Ülesannete klassifikatsioon tehniliste oskuste põhjal;
        1. iteratiivselt lahendatud probleemi rekursiooniga lahendamine;
        2. rekursiivselt lahendatud probleemi iteratiivselt lahendamine;
        3. puurekursioon (sh. binaarne rekursioon);
        4. lineaarne- ja sabarekursioon;
        5. mitmekordne rekursioon;

6. mitmepoolne rekursioon;
7. mõttes etteantud funktsiooni järgi avaldise väärtustamine olenevalt sisendist;
8. baasjuhu ja sammu äratundmine ning mõistmine;
9. alamülesandest terve ülesande lahenduseni jõudmine;
10. rekursiooni vähenemine baasjuhu suunas;
11. funktsiooni käitumine ootamatute argumentide korral;
12. rekursiivse funktsiooni olemus (nt. rekursiivne funktsioon ei pea alati midagi tagastama ega edastama);
13. rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast;
14. koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine;
15. algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek;
16. fikseeritud tasemete arvuga andmestruktuuri töötlemine;
17. fikseerimata tasemete arvuga andmestruktuuride töötlemine;
18. rekursiooni mälukasutus;
19. rekursiooni ajaline keerukus;
20. iteratiivse ja rekursiivse koodi ajalise keerukuse vahe;
21. iteratiivse ja rekursiivse koodi mälukasutuse vahe;
22. aktiveerimiskirjete ja/või rekursioonipuu koostamine;
23. rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine;
24. madal rekursioon;
25. sügav rekursioon;
26. rekursiivsed generaatorid;

- Tähelepanekuid ja häid ideid.

Töös kasutatud graafides ja analüüsis viidatakse edaspidi mallil nähtavatele 26-le tehnilisele oskusele kasutades nende järjekorranumbreid mallis. Tehnilised oskused ei ole mallil järjestatud ei paremuse ega olulisuse järgi.

4. Rekursiooni õpetamine Tartu Ülikoolis

Arendamaks käesoleva töö abil TÜ Informaatika õppekavas rekursiooni käsitlust, peab esmalt rahvusülikooli õppemeetodeid põhjalikult analüüsima ja mõistma. Alles märgates potentsiaalseid kitsaskohti, saab pakkuda tuttavale käsitlusele parandusi.

Bakalaureusetaseme Informaatika õppekavasse 2018/2019. sisse astunud tudengitele on kohustuslikud järgmised rekursiooni sisaldavad õppeained [9]:

1. LTAT.03.001 Programmeerimine (1. semester) (edaspidi LTAT.03.001);
2. LTAT.03.005 Algoritmid ja andmestruktuurid (3. semester) (edaspidi LTAT.03.005);
3. LTAT.03.006 Automaadid, keeled ja translaatorid (4. semester) (edaspidi LTAT.03.005).

Kokku käsitletakse rekursiooni kolmes TÜ informaatikutele kohustuslikus aines. Töö täielikkuse eesmärgil uuriti esmalt teemakäsitlust üle ülikooli st. võeti arvesse kõiki õppekavasi ja –astmeid (vt. täpsemalt Lisa I). Konkreetsuse ja tulemuslikkuse eesmärgil keskendub töö aga spetsiifiliselt TÜ bakalaureusetaseme Informaatika õppekava rekursiooni käsitluse parendamisele. Tasub mainida, et progressioon ainete LTAT.03.001 ja LTAT.03.005 vahel ning kõik järgnevalt väljatoodu mõjutab (olenevalt suunamooduli valikust) muu hulgas arvutitehnika, matemaatika ja matemaatilise statistika õppekava tudengeid (vt. täpsemalt Lisa I). Seega on töö tulemil potentsiaali ka nimetatud õppekavade parendamiseks.

4.1 Õpiväljundid

Rekursiooni esmaselt käsitlev aine LTAT.03.001 „ei nõua eelteadmisi peale üldise arvutikasutusoskuse” [14]. Järgmine suunitletud rekursiooni käsitlus on bakalaureusetaseme informaatikutele kolmandal semestril aines LTAT.03.005, mille eeldusaineks on rekursiooni mittekäsitlev õppeaine LTAT.03.003 „Objektorienteeritud programmmeerimine“ (edaspidi LTAT.03.003). Rekursiooniga seonduvad eelteadmised, mida ained LTAT.03.005 ja LTAT.03.006 tudengitelt ootavad, ei ole defineeritud.

Aine LTAT.03.001 õpiväljundites mainitakse rekursiooni põhiliste programmeerimiskonstruktsioonide hulgas, mida kursuse läbinud üliõpilane tundma ja kasutada oskama peab. Samuti kirjutatakse, et peale kursuse läbimist oskab üliõpilane „analüüsida ja üksikasjalikult selgitada programmi töö käiku”, seda „muuta, täiendada ja edasi

arendada” ning „luua lihtsamale ülesandele vastava algoritmi, koostada ja korrektselt vormistada lahendusprogrammi” [14].

Aine LTAT.03.005 õpiväljundites ei mainita rekursiooni. Teisalt on aine viienda loengu sisuks „tsükkel, rekursioon, magasin, järjekord” ja „nendevahelised seosed” ning osa kuuendast loengust käsitleb rekursiooni „eemaldamist” [15].

Õpiväljundites on kirjas, et kursuse läbinud tudeng oskab „võrrelda algoritmide töökiirust keerukushinnangute alusel”, „põhilisi andmestruktuure ja algoritme arvutis realiseerida, kasutada ja sobivamaid praktikas eelistada” [15]. Aines käsitletavad algoritmid nõuavad realiseerimiseks tihti rekursiooni. Seega võib järeldada, et nimetatud õpiväljundid – algoritmide keerukushinnangud, realiseerimine ja eelistamine – laienevad ka rekursioonile.

Aine LTAT.03.006 õpiväljundites ei mainita samuti rekursiooni. Aine ajakava lühikirjeldus annab märku, et aine kuuendas loengus käsitletakse vasakrekursiooni elimineerimist [16].

Õpiväljundite ja ainekavade põhjal saab järeldada, et kolme ainet läbinud tudeng peaks omama järgmisi teadmisi rekursioonist:

1. rekursiooni mõiste tundmine;
2. rekursiooni realiseerimine, kasutamine;
3. rekursiivse funktsiooni analüüsimine ja töö käigu selgitamine;
4. rekursiivse funktsiooni muutmine, täiendamine ja edasiarendamine;
5. rekursiivse funktsiooni efektiivsuse võrdlemine teiste lahendusmeetoditega ja eelistuse tegemine;
6. (vasak)rekursiooni elimineerimine.

Aine LTAT.03.006 seitsmenda praktikumi materjalides väidetakse, et kui tudeng saab aru, „kuidas vasakrekursiooni elimineerimise järel peab tagastama esialgse (vasak-assotsiatiivse operaatorite puhul) parsepuu”, siis võib teda „turvalise tarkvara arendamisega usaldada” [17]. Antud väite kaalu arvestades võib väita, et selleks, et tudeng oskaks rekursiooni oskuslikult rakendada hiljem väljaspool klassiruumi, on tarvis kõigi kuue taseme teadmisi. Selle saavutamiseks on omakorda oluline, et teadmuse õpetamise üleminek ainete vahel oleks taktikaliselt koordineeritud.

Aines LTAT.03.001 defineeritud mõiste *rekursioon* korratakse aine LTAT.03.005 materjalides üle koos õpitud mõistega *unaarne* ehk *lineaarne funktsioon*. Rekursiooniga

otseselt seotult tuuakse sisse ja kirjeldatakse uusi mõisteid *sabarekursioon* ja *rekursioonipuu* [18]. Huvitaval kombel on õppeaine materjalides on kirjas, et lugejalt eeldatakse „Python-põhise generaatori mõiste tundmist," [18] mida varasemates ainetes käsitletud ei ole.

Aine LTAT.03.006 rekursiooni mõistet ega põhimõtteid üle ei korda. Viienda õppenädala praktikumimaterjalide väide, et „puu töötlemine rekursiooniga ei ole ainult niisama harjutus, vaid selles peitub kogu Tõde programmeerimise kohta" [23] viitab aga sellele, et aine hindab rekursiooni oskust kõrgelt.

4.2 Õppematerjalid

Vaadeldavad ained tõid välja vähem kohustuslikke kui soovituslike õppematerjale. Töös on arvesse võetud kohustuslikke materjale.

Aine LTAT.03.001 kohustuslikuks materjaliks on e-õpik [13].

Aine LTAT.03.005 materjaliks on A. Pederi, J. Kiho ja H. Nestra „Algoritmid ja andmestruktuurid. Ülesannete kogu" (ISBN: 978-9949-77-377-0). Erandina on töösse kaasatud aine õpiväljundites mittekohustuslik A. Pederi ja H. Nestra materjal „Algoritmid ja andmestruktuurid. Tahvlipraktikum" kuna aine läbiviija rõhutas selle olulisust.

Aine LTAT.03.006 kohustuslikuks kirjanduseks on T. Æ. Mogenseni „*Introduction to Compiler Design*" (ISBN: 978-0-85729-829-4) ja R. Wilhelmi ja H. Seidli „*Compiler Design: Virtual Machine"* (ISBN: 978-3642149085).

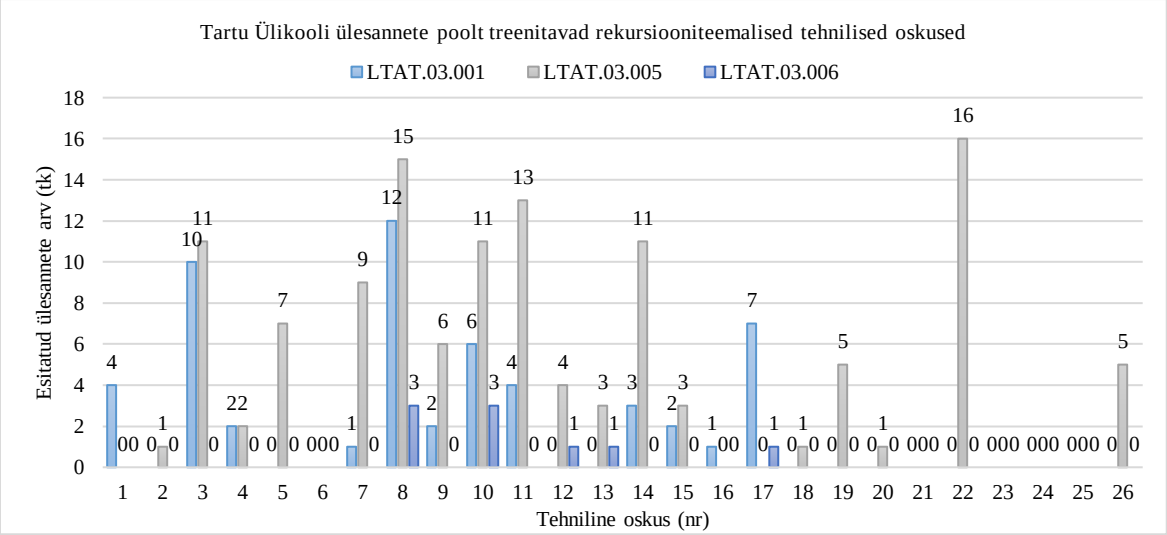
Õppeainete materjalid ei kordu ega viita otseselt sujuvale ainetevahelistele üleminekutele.

4.3 Tehnilised oskused läbi esitatud ülesannete

Tehnilised oskused

Rekursiooni tutvustakse TÜs hiljem kui iteratiivset programmeerimist. Üheks põhjuseks võib olla tõsiasi, et esimesel kokkupuutel rekursiooniga kasutatakse aines LTAT.03.001 õpetamismeetodit, mille raames lahendatakse varem iteratiivselt lahendatud probleemid võrdluse eesmärgil rekursiivselt.

Mõistmaks, millistele tehnilistele oskustele TÜ ainete esitatud ülesanded enim rõhku panevad, saab neid analüüsida kogumina.



Joonis 3. Tartu Ülikoolis esitatud ülesannete poolt treenitavad rekursiooniteemalised tehnilised oskused.

Joonis 3 illustreerib, millistele tehnilistele oskustele TÜ ainetes enim tähelepanu pööratakse. Selgub, et kumulatiivse ülesannete arvu põhjal peavad TÜ ained olulisimaks järgmist kuute tehnilist oskust:

1. (8) baasjuhu ja sammu äratundmine ning mõistmine;
2. (3) puurekursioon (sh. binaarne rekursioon);
3. (10) rekursiooni vähenemine baasjuhu suunas;
4. (11) funktsiooni käitumine ootamatute argumentide korral;
5. (22) aktiveerimiskirjete ja/või rekursioonipuu koostamine;
6. (14) koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine.

Võrreldes kuute jooniselt 3 selgunud olulist tehnilist oskust kuue õpiväljunditest tulenevaga, siis kattuvad neist neli esimest. Saadaval ei olnud aga ülesandeid, mis otseselt arendaksid oskust võrrelda rekursiivse funktsiooni efektiivsust teiste meetoditega, teha vastavalt eelistusi ega (vasak)elimineerida vasakrekursiooni. Võib järeldada, et esitatud ülesanded katavad mingil määral õpiväljundites määratud eesmärgid.

Töö uurib ühe osana, kuidas tudengeid aineks AA paremini ette valmistada. TÜs on enne AAd vaid üks aine, „Programmeerimine“, milles väärtustatakse ülesannete arvu põhjal enim järgmisi kuute tehnilist oskust: (8) baasjuhu ja sammu äratundmist ning mõistmist, (3) puurekursiooni (sh. binaarne rekursioon) tundmist, (17) fikseerimata tasemete arvuga andmestruktuuride töötlemist, (10) rekursiooni vähenemist baasjuhu suunas, (1)

iteratiivselt lahendatud probleemi rekursiooniga lahendamist ja (11) funktsiooni käitumise testimist ootamatute argumentide korral.

Oskused (8), (11), (3) ja (10) kattuvad muu hulgas aine LTAT.03.005 st. AA olulisimate oskustega. AA kursuses väärtustatud oskusteks, mida aine „Programmeerimine“ ei kata, on (14) koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine ja (22) aktiveerimiskirjete ja/või rekursioonipuu koostamine.

Kolme ainet silmas pidades selgub, et enim peetakse vajalikuks ja keskendutakse rekursiooni käsitluse algelisemat oskuse, baasjuhu ja sammu äratundmise ning mõistmise ja treenimisele. Oskuste ülekandumine ja kordumine osutab osati positiivsele tendentsile, et tudengid saavad õpitut korrata ja seeläbi kinnitada. Teisalt võib põhjus peituda ka selles, et ainete LTAT.03.005 ja LTAT.03.006 alguseks on rekursiivsed teadmised ununenud ning vajavad üle kordamist.

Samuti selgus, et TÜ ainete ülesanded ei katnud järgnevaid tehnilisi oskusi:

1. (6) mitmepoolne rekursioon;
2. (21) iteratiivse ja rekursiivse koodi mälukasutuse vahe;
3. (23) rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine;
4. (24) madal rekursioon;
5. (25) sügav rekursioon.

Esitatud ülesanded

Töö üheks põhiküsimuseks on ülesannete leidmine, mis annavad märku rekursiooni kasutamise vajadusest. Esitades selliseid ülesandeid tudengitele, saaks neid treenida intuitiivsemalt ära tundma olukordi, mil rekursioon on otstarbekas. Kolmest ainest ainsana esitab sedalaadi ülesandeid aine LTAT.03.005 (mittekohustuslik) tahvlipraktikumi materjal.

Esiteks kirjeldab materjal, et vajadus binaarse rekursiivse funktsiooni või protseduuri koostamise järgi „võib otseselt tuleneda nii ülesande püstitusest (nagu nt. Fibonacci jada) kui ka asjaolust, et muul viisil (nt. iteratiivsel moel) on antud ülesande lahendamine oluliselt keerukam” [18:26]. Teiseks toob materjal kontrastina välja olukorra, millal rekursiooni kasutamine ei ole ratsionaalne. Õppematerjal juhib tähelepanu, et multirekursiooni ei ole mõistlik kasutada praktilises programmeerimistöös. Selgitusena toob materjal, et lahendades permutatsioonide ja kombinatsioonide ülesandeid

multirekursiooniga on „tegemist on õppeotstarbelise suunitlusega aruteludega” [18:34] ning, et „praktilises programmeerimistöös on mõistlik kasutada vastavaid permutatsioonide ja kombinatsioonide tarkvaralisi generaatoreid, Pythonis näiteks teegis *itertools* leiduvaid vahendeid” [18:36]. Samuti mainib aine neljas loeng, et rekursioon ei sobi, kui tahame midagi teha „lõpmatult” palju [24].

4.4 Õpetamismeetodid

Ained LTAT.03.001 ja LTAT.03.006 panevad rekursiooni teooriale vähe rõhku ja piirduvad suuresti õpetamisega näidete põhjal. Aine LTAT.03.005 läheneb aga olukorrale mitmeti. Tahvlipraktikumis mainitakse, et rekursiooni juures on oluline „koostatava, antud ülesannet lahendava funktsiooni nn väline spetsifitseerimine: tuleb täpselt iseloomustada parameetrite otstarve ja samuti kirjeldada oodatav tulemus” [18:19]. Seejärel kohtab tudeng kolme selgitust, kuidas rekursiooniga end kodusemalt tunda. Esiteks soovitatakse tahvlipraktikumis skeemi koostamist, „mis väljendab „tavaelule sarnanevat „kõikide variantide süsteemset ülevaatust nende töötlemise järjekorras” [18:26]. Materjal soovitab seda, et skeemist lõpuks rekursioonipuu tekitada, mida tööprotsessi (rekursioonipuud) jälgides omakorda rekursiooniga osadeks jaotada. Rekursioonipuu konstrueerimisel soovivad kolmas ja neljas loeng [24, 25] mõelda „rekursiooni disainist kui puu osadest”: juur st. lähtekoht; lehed st. osad, mida enam ei jaotata; vahetipud st. uus rekursiivne funktsioonikutse ja servad st. alamprogrammide funktsioonikutsete ja tagastuste analüüs. Soovitatakse veel järgnevaid samme [25]:

1. Leia, kuidas ülesannet lihtsamateks alamülesanneteks jagada.
2. Leia, kuidas tuletada alamülesannete lahendustest käesoleva ülesande lahendus.
3. Leia lõpetamistingimus, mille poole jagamine viib.
4. Määra lahendus, kui lõpetamistingimus on täidetud.

Aine LTAT.03.005 väljapaistvaks õpetamismeetodiks on läbimänguslaidide kasutamine, et algoritmide tööd visualiseerida. Samuti õpetatakse neljandas loengus [24] mõningaid jaotamisprintsiipe, millega rekursiivse funktsiooni kirjutamist lihtsustada.

4.5 Järeldused

Tugevused

Ülesannete rohkus läbi ainete on igati positiivne, sest tudengil on seeläbi kohustus, ent ka võimalus, piisavalt harjutada. Aine LTAT.03.001 suureks tugevuseks on kättesaadav ja tagasisidestatav e-õpik. Samuti tähendavad e-õpiku peidetud vihjed ja näidislahendused, et tudeng saab vajadusel abi, kuid muu hulgas jooksvalt ise oma teadmisi kontrollida.

Spetsiifiline olukord, millal võib eksisteerida argumentideta rekursioon, on aine LTAT.03.001 e-õpikus efektiivselt põhjendatud: kui baas on defineeritud muutujast olenemata, siis „võib rekursioon toimuda ka ilma väärtusi edastamata" [13]. Sellise olukorra täpsustamine on oluline, sest annab tudengile märku rekursiooni mitmekülgsest olemusest.

Aine LTAT.03.005 tahvlipraktikumide failis on iga ülesande juures temaga seonduvad mõisted. Seda metoodikat võiks rakendada iga aine harjutustele, sest annab tudengile ülesande olemusest sügavama arusaama.

Kitsaskohad

Õppeainete õpiväljundites nimetatakse rekursiooni kas liiga napisõnaliselt või üldsegi mitte. See tekitab olukorra, kus aines õpitu ei pruugi kattuda sellega, milleks tudeng end ette valmistab või mida ta ainelt taotleb.

Kuna ülesandeid on kohati liiga palju, siis peaksid nad olema prioriseeritud. Ülesanne võiks tudengini jõuda koos sõnaselge nimekirjaga tehnilistest oskustest, mida ta treenib. Vastasel juhul võib tekkida olukord, kus tudeng ei oska valida olulisemaid ega piisavalt varieeruvaid harjutusi ja valib tõenäoliselt ajanappuses vaid mõned juhuslikud. Samuti kuluks igasse ainesse ära rohkem päriselu näiteid, kus mainitakse nt. rekursiooni rakendusi tarkvaraarenduses.

Aines LTAT.03.006 ei korrata rekursiooni mõistet üle. Julgen väita, et tegu on tehnikaga, mille põhitõdede kordamine toob märgatavalt rohkem kasu kui kahju. Aines LTAT.03.005 küll korratakse rekursiooni olemus üle, ent kõnealune on materjalis, mis ei ole ametlikult kohustuslik. Õppeainetel tasuks seega oma kohustuslike materjalide nimekirja üle vaadata.

Niisama tähtis on olulise väite juures näite toomine. Näide õppematerjalides esitatud väitest, mis vajaks illustratiivset näidet: „binaarse rekursiooni juhul, kus funktsiooni või

protseduuri kehas on ette nähtud rekursiivne funktsioonikutse kahel korral, ei pruugi (loomulikku) iteratiivset analoogi alati leidudagi" [18]. Lõpetuseks, tuues paralleele LTAT.03.001 e-õpikuga, sobiksid vihjed ja/ või osalised (kuid peidetud) lahenduskäigud ka LTAT.03.005 ülesannete kogusse.

Kokkuvõte

TÜ tudengid mõistavad rekursiooni keerukust. Kahjuks ei paku ülikool veel õpiväljundite põhjal hinnates täielikult koordineeritud käsitlust ega materjale. Kuna osad tehnilised oskused jäävad katmata, siis ei saa kindlusega esitatud ülesannete alusel öelda, et kolme aine läbimisega saavutatakse kõik õpiväljundi. Selgus, et enim esitatakse ülesandeid, mis keskenduvad rekursiooni käsitluse algelisema oskuse, baasjuhu ja sammu äratundmise, mõistmise treenimisele.

Kumulatiivse ülesannete arvu põhjal peetakse TÜs olulisimaks tehnilisi oskusi (8), (3), (10), (11), (22) ja (14). Katmata jäid aga oskused (6) mitmepoolne rekursioon, (21) iteratiivse ja rekursiivse koodi mälukasutuse vahe, (23) rekursiivse ja iteratiivse lahendus-meetodi vahel eelistuse tegemine, (24) madal rekursioon ja (25) sügav rekursioon.

TÜs on enne AAd vaid üks aine, „Programmeerimine", milles väärtustatakse ülesannete arvu põhjal enim oskusi (8), (3), (17), (10), (1) ja (11).

Olgugi, et esitatud ülesannete hulk on suur, siis ei tooda arvestataval määral välja ülesandeid, mis spetsiifiliselt vihjaksid rekursiooni kasutamise vajalikkusele. Ainsateks näideteks on käesoleva uurimuse põhjal spetsiifiline Fibonacci arvude kalkuleerimine kasutades binaarset rekursiooni ning tõsiasi, et rekursioon ei sobi ülesannete jaoks, mis nõuavad millegi lõpmatu kord tegemist.

Ei selgunud ülesandeid, mis otseselt arendaksid oskust võrrelda rekursiivse funktsiooni efektiivsust teiste meetoditega, teha vastavalt eelistusi ega (vasak)elimineerida vasak-rekursiooni. Ained LTAT.03.001 ja LTAT.03.006 panevad rekursiooni teooriale vähe rõhku ja piirduvad suuresti näidete põhjal õpetamisega. Siiski soovitatakse probleemilahendusmeetoditena mõelda „rekursiooni disainist kui puu osadest", ülesande jagamist lihtsamateks alamülesanneteks ning rekursioonipuu koostamist, „mis väljendab „tavaelule sarnanevat„ kõikide variantide süsteemset ülevaatust nende töötlemise järjekorras" [18:26].

5. Rekursiooni õpetamine teistes Eesti ülikoolides

Eesti ülikoolide valimisse kuulusid algselt Tallinna Tehnikaülikool (edaspidi TalTech), Tallinna Ülikool (edaspidi TLÜ) ja Eesti Infotehnoloogia Kolledž (edaspidi IT Kolledž). TLÜ Informaatika õppekava ühegi kohustusliku õppeaine õpiväljundid ega sisu rekursiooni ei maininud. Avaliku materjali puudumise tõttu ei ole võimalik TLÜd valimisse võtta. 2017. aastal ühines IT Kolledž TalTechi infotehnoloogia teaduskonnaga [26]. Seega koosneb töö Eesti ülikoolide valim vaid TalTechist.

TalTechi bakalaureusetaseme Informaatika õppekava (IAIB 17/19) tudengitele on kohustuslikud järgmised rekursiooni sisaldavad õppeained [27]:

- ITI0102 Programmeerimise algkursus (1. semester) (edaspidi ITI0102);
- ITI0202 Programmeerimise põhikursus (2. semester) (edaspidi ITI0202);
- ITI0204 Algoritmid ja andmestruktuurid (3. semester) (edaspidi ITI0204).

Sarnaselt TÜle käsitleb TalTech rekursiooni kolmes õppeaines. Peamiseks erinevuseks on TalTechi programmeerimiskursus, mis on jagatud kaheks osaks.

5.1 Õpiväljundid

Rekursiooni ei mainita õppekava ühegi aine väljundis ega sisukirjelduses. Seega saab teha järeldusi eeldustest ja õppekava läbimisega saavutatavast tasemest vaid esitatud ülesannete põhjal.

5.2 Õppematerjalid

Aine ITI0102 õppematerjalideks [28] on e-õpik [32] ning J. M. Zelle raamat „*Python Programming: An Introduction to Computer Science*“ (ISBN: 1590282752).

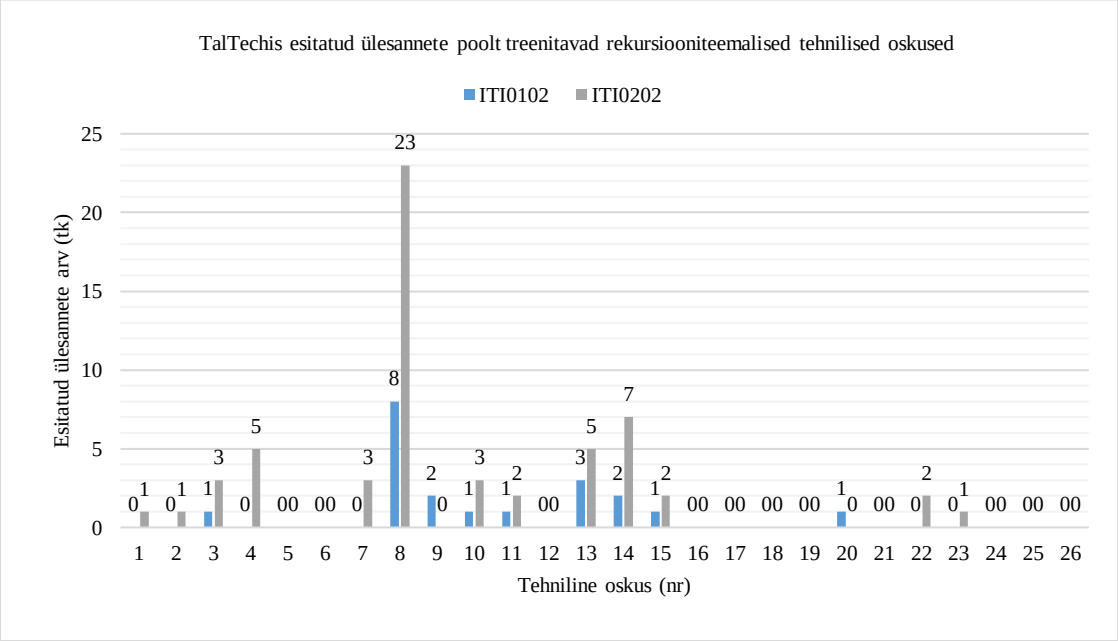
Aine ITI0204 õppematerjalide hulka kuuluvad kolm raamatut [29]: R. Neapolitani ja K. Naimipouri „*Foundations of Algorithms*” (ISBN: 0763782505), T. H. Cormeni, C. E. Leisersoni jt. „*Introduction to Algorithms*” (ISBN: 9780262033848) ning J. Kiho „Algoritmid ja andmestruktuurid” (ISBN: ISBN 9985-56-767-6). Ülejäänud õppematerjalidele ei võimaldatud liigipääsu.

TÜga kattuvaid õppematerjale ei ole. Kuna aine ITI0204 õppematerjalidele ei ole ligipääsu, siis keskendub edasine töö ainete ITI0102 ja ITI0202 käsitluse uurimisele.

5.3 Tehnilised oskused läbi esitatud ülesannete

Tehnilised oskused

Sarnaselt TÜle õpetab TalTech samuti rekursiooni hiljem kui iteratiivset programmeerimist. Ülesandeid, mida esitati, oli aga märgatavalt vähem (kättesaadaval).



Joonis 4: TalTechis esitatud rekursiooniteemalised ülesanded.

Joonis 4 illustreerib, millistele tehnilistele oskustele TalTechi õppeainetes enim tähelepanu pööratakse. Kumulatiivset ülesannete arvu arvesse võttes peetakse kahe kursuse peale kokku olulisimaks järgmist kuute tehnilist oskust:

1. (8) baasjuhu ja sammu äratundmine ning mõistmine;
2. (14) koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine;
3. (13) rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast;
4. (4) lineaarne- ja sabarekursioon;
5. (3) puurekursioon (sh. binaarne rekursioon);
6. (10) rekursiooni vähenemine baasjuhu suunas.

TÜs hinnati oskuste (13) ja (4) asemel oskusi (10) (rekursiooni vähenemine baasjuhu suunas ja) (11) (funktsiooni käitumine ootamatute argumentide korral). Olgugi, et TalTechi AA materjalid ei olnud saadaval, siis saab eelnevat kahte ainet põhjalikumalt vaadeldes uurida, milliste tehniliste oskustega tudengeid AAks ette valmistatakse.

Tabel 1: TalTechi õppeainete ITI0102 ja ITI0202 poolt olulisimaks peetud rekursiooniga seonduvad tehnilised oskused enne AA õppimist.

	ITI0102	ITI0202
1.	(8) baasjuhu ja sammu äratundmine ning mõistmine;	(8) baasjuhu ja sammu äratundmine ning mõistmine;
2.	(13) rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast;	(14) koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine;
3./ 4.	(9) alamülesandest terve ülesande lahenduseni jõudmine; (14) koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine;	(4) lineaarne- ja sabarekursioon; (13) rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast;
5.	Viik ülejäänud tehniliste oskuste (3), (10), (11), (15), (20) vahel.	Viik kolme tehniliste oskuse (3), (7), (10) vahel.

Tabelis 1 on helehalli taustaga tähistatud populaarsed tehnilised oskused, mis jätkuvad ainest ITI0102 ainesse ITI0202. Samas leidus suur hulk tehnilisi oskusi, mida kumbki aine läbi ülesannete ei õpetanud ja seega ka enne AAd hädavajalikuks ei pidanud:

1. (5) mitmekordne rekursioon;
2. (6) mitmeetapiline rekursioon;
3. (9) koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine;
4. (12) rekursiivse funktsiooni olemus;
5. (16) fikseeritud tasemete arvuga andmestruktuuri töötlemine;
6. (17) fikseerimata tasemete arvuga andmestruktuuride töötlemine;
7. (18) rekursiooni mälukasutus;
8. (19) rekursiooni ajaline keerukus;
9. (21) iteratiivse ja rekursiivse koodi mälukasutuse vahe;
10. (24) madal rekursioon;
11. (25) sügav rekursioon;
12. (26) rekursiivsed generaatorid.

Tehnilisteks oskusteks, mida TÜ oma esitatud harjutusega ei treeninud, aga TalTech treenis on (23) iteratiivse ja rekursiivse koodi ajalise keerukuse vahe ning (26) rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine.

Kohati läbimõtlemata tehniliste oskuste arengust TalTechi ainete vahel annab märku tõsiasi, et aine ITI0202 kordab enda e-õpikus kõiki näiteülesandeid, mida kursuse ITI0102 e-õpik juba eelneval semestril põhjalikult käsitles.

Esitatud ülesanded

Tutvusime eelmises peatükis TÜs väljapakutud harjutustega, mis andsid märku nii rekursiooni kasutamise vajadusest kui ka sellest, et vahel on targem rekursiooni vältida. TalTechi õppekava toob esmalt välja fraktaalid, mis annavad märku rekursiooni kasutamise vajalikkusest. Õppeaine ITI0102 õpik [19] esitab harjutused kirjutamaks funktsioonid Kochi lumehelbe ja C-kurvi visualiseerimiseks. Ülesannete püstitustes vihjatakse, et rekursiivne lahendus aitab vältida koodi kordust.

Õppeaine ITI0202 õpiku [10] ülesanded sisaldavad samamoodi fraktaale: kolme erinevat harjutust Sierpinski kolmnurgaga, H-puud, Hilberti kurvi ja rekursiivset puud. Üleüldse viidatakse eraldi andmetüüpidele, nt. kaustapuudele, mis on loomult rekursiivsed.

Kaustapuude osas pakub õpik [10] välja viis erinevat ülesannet. Näiteks peab puurekursiooniga leidma kausta failide arvu, sõna esinemise arvu otsides etteantud kausta kõikidest failidest ja asendama kindel sõna etteantud kausta kõikides failides. Edasijõudnute harjutus on kausta suuruse leidmine ilma rekursioonita.

Samuti oodatakse tudengilt iteratiivse ja rekursiivse lahendusmeetodi vahel eelistuse tegemist. Aine õpik selgitab, et sabarekursioon on ihaldusväärne lahendusmeetod kuna meetod lõpeb samaaegselt, kui rekursiivne funktsioonikutse ning seega ei ole vajadust hoiustada vahepealseid funktsioonikutseid magasinis. Nimetatud selgitus annab tudengile märku efektiivsemast mälukasutusest. Veelgi enam mainib õppematerjal, et kompilaatorid saavad sabarekursiooni optimeerida, et magasini suurust (ja sh. mälukasutust) vähendada.

5.4 Õpetamismeetodid

TalTechi tudengeid õpetatakse mõlemas aines läbi klassikaliste loengute ja praktikumide. Aine ITI0102 kavas on veel tunnikontrollid [30] ning aine ITI0202 sisaldab muu hulgas projektipraktikume ja konsultatsioone [31].

Huvitava tähelepanekuna saab tuua võrdluse, et kui rekursiooni õpetati TÜ programmeerimiskursuse lõpus, siis ITI0102 käsitleb rekursiooni semestri keskel st. 9. nädalal.

TalTech õpetab enda tudengitele rekursiooni aines ITI0102 alustades mõiste selgitamist hoopis matemaatikast: „matemaatikas kasutatakse rekursiooni rekurrentsetes arvujada-

des. Arvujada on rekurrentne siis, kui jada mingi väärtus sõltub eelmisest väärtusest" [32]. Õppeaine ITI0202 toob aga päriselulisi näiteid, mis illustreerivad rekursiooni mõistet. Näiteks illustreeritakse baasi ja sammu kohvijoomise (st. joomine toimub kuni tass on tühi) ning *n*–korda sõnumi printimisega [10].

Kursused toovad välja mitmeid viise, kuidas nutikalt rekursiivset ülesannet lahendada. Näiteks tõi aine ITI0102 e-õpik välja meetodi proovida enne „määrata tingimusi, kui rekursiooni ei pea kasutama" [32]. Ülesannete lahendamiseks, mis sarnanevad alamsõne indeksi sõnest otsimise ülesandega, soovitab sama e-õpik lahendusmeetodina järgmist algoritmi [32]:

1. Jaga sõna kaheks osaks. (Osa suurus sõltub ülesandest.)
2. Vaata, kas esimene osa täidab vajalikku nõuet.
    - Kui jah, siis tagasta tulemus.
    - Kui ei, siis uuri teist osa selle algoritmi järgi veel kord (rekursioon). Esimene osa jäta muutmata.

Aine ITI0202 õpik oli aga abiks tudengile, kes on raskuses baasi ja sammu defineerimisega. Õpik kirjeldab [10], et lihtsaim viis teha kindlaks, et tegu on korrektse sammu ja baasiga, on omakorda teha kindlaks, et iga rekursiivne funktsioonikutse toimib originaalsest probleemist väiksemal probleemil. Täpsemalt peab samm iga rekursiivse funktsioonikutsega probleemi baasjuhule lähendama. Õpik sõnastab, et baasjuhuks saab väike osa tervest probleemist, mida saab lahendada rekursioonita. Samuti täpsustatakse, et baasjuhte, mis rekursiooni peatavad, võib olla mitu.

Huvitavaks lahendusmeetodiks on kitsendus, mille raames aine ITI0102 ei soovita rekursiooni Pythoniga koos kasutada. Väidetakse, et: „Pythonis on rekursiivne lahendus alati ebaefektiivsem kui analoogne iteratiivne lahendus" [33]. Aine ITI0202 toob sarnase olukorra välja Java kohta väites, et "Java keel kahjuks ei oska rekursiooniga optimaalselt hakkama saada. Seega, rekursiivne funktsioonikutse on tavaliselt natuke aeglasem kui iteratiivne. See vahe on minimaalne (või olematu). Pigem valitakse rekursiivne lahendus seetõttu, et see on lühem ja selgem" [34]. Rekursiooni õpetatakse antud ainetes aga just keeltega Python ja Java. Siinkohal tasuks tuua mõni näide, harjutus või viide, mille abil tudeng saaks edasi töötada keelega, mis saab rekursiooniga optimaalselt hakkama või mainida edasist kursust (üldiselt seotud funktsionaalsete programmeerimiskeeltega),

milles tudeng õpitud oskusi täie kindlusega rakendada saaks. Seeläbi oskaks tudeng endale paremini lahti mõtestada, miks ja millal rekursiooni on tarvis.

Küsimusi „miks?“ ja „millal?“ illustreerib hästi aine ITI0102 õpik [19], kui võrdleb rekursiooni iteratsiooniga. Õpik toob välja, et rekursiivsed funktsioonid on tsüklite üldistused ning kõik, mida saab teha tsükliga, saab teha ka lihtsa rekursiivse funktsiooniga. Eripärana nimetati tõsiasi, et leidub programmeerimiskeeli, mis kasutavad eksklusiivselt ainult rekursiooni. Samuti mainib õpik motiveerivalt, et on ülesandeid, nt. Hanoi tornid, mida on rekursiivselt peaaegu triviaalne, ent iteratiivselt väga keeruline, lahendada.

Õpik [19] hoiatab tudengeid väga ebaefektiivsete rekursiivsete algoritmide, nagu rekursiivselt Fibonacci jada elemendi leidmine, osas. Õpikuväliselt võrdleb aine e-õpik [32] sama Fibonacci ülesande rekursiivset ja iteratiivset lahendust. Silma paistab meetod, mille raames parandatakse rekursiivset lahendust iteratiivse heade külgede põhjal st. lisatakse mälu tehes lahendus seega iteratiivsest kiiremaks. Näitena sooritatakse veel kiirusvõrdlusi, mis on abistavaks lahendusmeetodiks tudengile, kes kahtleb oma rekursiivse ja iteratiivse lahenduse vahel.

Viimasena nendib aine ITI0202 õpik, et rekursioon „nõuab harjutamist enne, kui see mugavaks muutub“ [19:481]. Õpik kinnitab õppurile: „kuni sa jälgid reegleid ja teed kindlaks, et iga rekursiivne funktsioonikutsete "kett" jõuab lõpuks baasi, siis sinu algoritmid töötavad“ [19:481].

5.5 Järeldused

Tugevused

TalTechi mõlema uuritud aine põhitugevuseks on põhjalikud e-õpikud, mis annavad tudengile võimaluse enda rekursioonialaseid teadmisi vajadusel meelde tuletada ka peale kursuste läbimist.

Aine ITI0102 tugevuseks oli osade e-õpikus [32] toodud näidete (nt. faktoriaal ja alamsõne indeksi leidmine sõnes) aeglaselt ja põhjalikult selgitatud lahendusmeetodid. Samuti kasutati omapärast teguviisi, mille raames võrreldi iteratiivset ja rekursiivset Fibonacci funktsiooni ning lisati rekursiivsele peale sooritusvõrdlusi mälu. Kahjuks tehti antud võrdlust ühekordse näitena. Uudne lahendus oleks sellist lähenemist ka rohkemates esitatud ülesannetes rakendada.

Aine ITI0202 e-õpikus [34] käidi uuesti üle rekursiooni olemus ja mõiste. See aitab tudengeid võrdsesse seisu isegi siis, kui aine ITI0102 möödudes on rekursioon ununenud. Õppeaines toodi päriselulisi näiteid, mis on loomult abiks keerukamate ideede, nagu rekursiooni, olemuse kinnistamisele. Samuti olid ülesanded märgitud tärniga vastavalt nende raskusastmest. Selline teguviis annab tudengile märku, et vaid lihtsamate ülesannete lahendamisest ei pruugi piisata ning aitab valida mitmekesist ülesannete komplekti juhul, kui kõik ülesanded ei ole kohustuslikud. Peale selle lahendati veel sama probleemiga ülesandeid kasutades mitmeid andmetüüpe (nt. sõne, massiiv, täisarv).

Sarnaselt ainele ITI0102 on ines ITI0202 väga põhjalik ning aeglaselt selgitatud lahendusmeetod ülesandele, mis otsis rekursiivselt sõnest alamsõne indeksit. Täpsustame, et täpselt sama ülesannet lahendati juba kord üksikasjalikult aines ITI0102.

Aine suureks tugevuseks on veel see, et õpiku iga ülesanne lõppes käsuga „kirjuta testprogramm, mis ootab kasutajalt...” [19:497]. Selline lähenemine treenib tudengit muu hulgas arvesse võtma erijuhte ja õpetab hinnalist oskust, milleks on oma koodi testimine.

Kitsaskohad

Rekursiooni mainitakse TalTechi ainete õpiväljundites samuti liiga vähe st. antud ülikooli näitel mitte kordagi. Napisõnalised õpiväljundid ja ainekirjeldused jätavad mulje, et aineid ja õppekava läbides ei peagi rekursiooni omandama.

Niisama oluline on tõsiasi, et aine ITI0202 e-õpikus ei olnud ühtegi lahenduseta ülesannet ning aine ITI0102 e-õpikus oli vaid üks. Veelgi enam, aine ITI0202 kasutas samu näiteid, mida eelnevas kursuses juba põhjalikult läbiti. Siinkohal tasuks neid vanu näiteid küll kasutada, aga ehk hoopis täiendatud ülesannete vormis.

Sarnaselt TÜ AA kursusega esitas aine ITI0202 õpik kohati liiga palju ülesandeid ning ei täpsustanud nende eesmärke. Olgugi, et täpselt ei ole määratletud, kas tudengitelt nõutakse kõikide ülesannete lahendamist, siis oleks hea praktika täpsustada, millised ülesanded on olulisimad ning milliseid tehnilisi oskusi nendega arendatakse.

Kokkuvõte

Sarnaselt TÜle käsitles TalTech rekursiooni kolmes õppeaines. Peamiseks erinevuseks oli TalTechi programmeerimise kursus, mis on jagatud kaheks osaks. Samuti, kui rekursiooni õpetati TÜ programmeerimiskursuse lõpus, siis ITI0102 käsitles rekursiooni semestri keskel.

Kumulatiivselt hinnati enim oskusi (8), (14), (13), (4), (3) ja (10). Kui TÜs hinnati oskusi (10) rekursiooni vähenemine baasjuhu suunas ja (11) funktsiooni käitumine ootamatute argumentide korral, siis TalTechis väärtustati oskusi (13) rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast ja (14) rohkem koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine. Veelgi enam, kui TÜs ei treenitud oskusi (23) iteratiivse ja rekursiivse koodi ajalise keerukuse vahe ja (26) rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine, siis TalTechis treeniti. Selgus, et TÜ pidas enne AA õppimist olulisemaks rohkem tehnilisi oskusi, kui TalTech.

Fraktaale ja rekursiivseid andmetüüpe, nagu kaustapuid, toodi välja kui probleeme, mis annavad märku rekursiooni kasutamise vajalikkusest. Tudengitele esitati ülesanne rekursiivse andmestruktuuriga, mille pidi iteratiivselt lahendama, et rõhutada rekursiooni paremat sobivust tollel juhul. Samuti käsitleti sabarekursiooni soodsust ja ihaldatavust.

Kokkuvõtlikult esitasid TalTechi õppeained küll vähem ülesandeid, kuid lähenemine tundus täpsem. Näiteks küsiti sama probleemi lahendust mitmes erinevas programmeerimiskeeles. Samuti toodi päriselulisi näiteid ja anti soovitusi, nagu tingimuse määramine, millal rekursiooni ei pea kasutama. TalTechi üks õpikutest hoiatas tudengeid ka väga ebaefektiivsete rekursiivsete algoritmide, nagu *n*-is Fibonacci number, osas. Näitena sooritati ka ajalisi sooritusvõrdlusi, mis on abiks tudengile, kes kahtleb probleemi rekursiivse ja iteratiivse lahenduse vahel.

6. Rekursiooni õpetamine välisülikoolides

6.1 Ülikoolide valim

Välisülikoolide valimisse valiti ülikoole järgnevalt:

- Ülikoolid, mis on TÜ arvutiteaduse instituudi Erasmus üliõpilasvahetus-programmi partnerülikoolid [36].
- Tartu Ülikool on 2019. aastal *Times Higher Education* portaali [35] sõnul maailmas paremuselt 251−300s informaatikat õpetav ülikool. Väisülikoolide valimisse valiti nii ülikoole, mis olid rahvusülikooliga sarnasel tasemel, kui ka ülikoole, mis olid edetabelis eespool. Seeläbi saab töö kajastada samalaadsete ülikoolide meetodeid, aga parendada ka rahvusülikooli edukate näitel.
- Olgugi, et esialgne valim oli mahult kogukas, siis selgus, et oli üllatavalt vähe ülikoole, mille materjalid olid avatud, põhjalikud ja asjakohaseid õppeaineid läbivad. Seetõttu valiti valimisse lisaks ülikoole, mille avatud materjalidele uurimuse käigus mõni juba valimis olev ülikool suunas.

Lõplik valim koosneb kümnest põhjalike materjalidega ülikoolist ning kuuest ülikoolist, mille õppeained olid töö kontekstis kasutoovalt defineerinud enda õppeväljundeid ja/ või materjale. Viimased on võetud valimisse eesmärgiga leida asjakohast kirjandust ja näha, milliseid tehnilisi oskusi suurem hulk ülikoole tähtsaks peab.

Ülikoolid põhjalike materjalidega:

1. Rensselaeri Polütehniline Instituut (ingl *Rensselaer Polytechnic Institute*, ülikoolide paremusjärjestuses kohal 301-400 [35], edaspidi RPI*);*
2. Bergeni Ülikool (ingl *University of Bergen*, ülikoolide paremusjärjestuses kohal 201-250 [35]);
3. Kataloonia Polütehniline Ülikool (ingl *Polytechnic University of Catalonia*, ülikoolide paremusjärjestuses kohal 126-150 [35], TÜ Erasmus partner [36], edaspidi KPÜ);
4. Helsinki Ülikool (ingl *University of Helsinki*, ülikoolide paremusjärjestuses kohal 126-150 [35], TÜ Erasmus partner [36]);
5. Chalmersi Ülikool (ingl *Chalmers University*, ülikoolide paremusjärjestuses kohal 101-125 [35]);

6. Aalto Ülikool (ingl *Aalto University*, ülikoolide paremusjärjestuses kohal 88 [35], TÜ Erasmus partner [36]);
7. Marylandi Ülikool, College Park (ingl *University of Maryland, College Park*, ülikoolide paremusjärjestuses kohal 44 [35]);
8. California Ülikool, Berkeley (ingl *University of California, Berkeley*, ülikoolide paremusjärjestuses kohal 11 [35]);
9. Princetoni Ülikooli (ingl *Princeton University*, ülikoolide paremusjärjestuses kohal 8 [35]);
10. Massachusettsi Tehnoloogiainstituut (ingl *Massachusetts Institute of Technology*, edaspidi MIT, ülikoolide paremusjärjestuses kohal 4 [35], edaspidi MIT).

Ülikoolid heade õppematerjalidega ja/või eeskujulike õpiväljunditega:

1. Bremeni Ülikool (ingl *University of Bremen*, ülikoolide paremusjärjestuses kohal 251-300 [35], TÜ Erasmus partner [36]);
2. Johannes Kepleri Ülikool Linzis (ingl *Johannes Kepler University of Linz*, ülikoolide paremusjärjestuses kohal 251-300 [35]);
3. Konstanzi Ülikool (ingl *University of Konstanz*, ülikoolide paremusjärjestuses kohal 201-250 [35], TÜ Erasmus partner [36]);
4. Genti Ülikool (ingl *Ghent University*, ülikoolide paremusjärjestuses kohal 201-250 [35]);
5. Ulmi Ülikool (ingl *Ulm University*, ülikoolide paremusjärjestuses kohal 176-200 [35], TÜ Erasmus partner [36]);
6. Aalborgi Ülikool (ingl *Aalborg University*, ülikoolide paremusjärjestuses kohal 151-175 [35]).

6.2 Rekursiooni käsitlevad õppeained

Valimi kuusteist ülikooli käsitlesid kaudselt või konkreetselt rekursiooni kokku 48s õppeaines.

Kõige rohkem rekursiooni käsitlevaid õppeaineid (5) oli RPIs ja Aalborgi Ülikoolis. Kõige vähem leidus materjalidega rekursiooni käsitlevaid aineid (1) Helsinki Ülikoolis. Nii aritmeetiline keskmine kui ka mediaan õppeainete arvu suhtest ülikooli kohta oli 3.

Tabel 2. Maailma ülikoolide rekursiooni käsitlevate õppeainete jaotus aine- ja semestripõhiselt.

Aine/ Semester	1.	2.	3.	4.	5.	6.
„Arvutiteaduse alused“	xxxxx		x			
„Teoreetiline arvutiteadus“	x			x		
„Programmeerimine“	x					
„Programmeerimine I“	xx					
„Programmeerimine II“			xxx			
FP ehk „Funktsionaalne programmeerimine“	x		xx	x		
OOP I ehk „Objektorienteeritud programmeerimine I“	xx					
OOP II ehk „Objektorienteeritud programmeerimine II“		xx				
„Programmeerimiskeeled“				x		x
AA ehk „Algoritmid ja andmestruktuurid“		xxxx	xxxxxx			
„Algoritmid“				xx	xx	
„Andmestruktuurid“		xx		x		
„Tõenäosus ja keerukus“			x	x		
„Arvutiprogrammide struktuur ja tõlgendamine“	x					
„Loogika ja algebra“	x					
„Süntaks ja semantika“				x		
„Modelleerimine ja verifitseerimine“						x

Tabelis 2 tähistab iga „x“ ühte õppeainet. Tabel 2 illustreerib, et enim õpetatakse rekursiooni 1. semestril (14 õppeainet) ja 3. semestril (13 õppeainet). Enim käsitlevad rekursiooni ained AA (10 ainet), „Arvutiteaduse alused“ (6 ainet), FP (4 ainet) ning „Algoritmid“ (4 ainet).

TÜ AA õppeaine eest vastutav õppejõud tõi välja hüpoteesi, et TÜs peaks enne ainet AA olema rohkem kui üks rekursiooni tutvustav kursus, milleks on praegu kõige esimene programmeerimiskursus. TÜga sarnane struktuur (st. vaid üks aine enne AAd) on välisülikoolide valimi põhjal veel ainult Genti Ülikoolis (paremuselt 201.−250. ülikool), kuigi ülikool on kompenseerimiseks jaganud AA kaheks kursuseks [40].

Maailma ülikoole uurides selgub, et üldiselt on õppeained „Programmeerimine“ ja OOP omakorda kaheks jaotatud. Selgus, et TÜga sarnaselt käsitleb rekursiooni kas programmeerimiskursus(ed) või OOPi kursus(ed), aga mitte mõlemad. Veelgi enam, kui ülikool õpetab OOPi kahe eraldi ainena, siis tavalist programmeerimiskursust ei ole (ja vastupidi). Samas näiteks Bremeni (251.−300.) ja Bergeni (201.−250.) Ülikoolis käsitleti rekursiooni nii OOPi kui ka FP kursustes. Muu hulgas leidub olukordi, kus rekursiooni

õpetamine on FP kursuse näol isegi kolmeks jaotatud. Tasub mainida, et mitmetes ülikoolides (nt. RPI ja Gent) on AA õpetamine samuti kahe eraldiseisva kursuse peale jaotatud.

Valimi õppekavade põhjal ilmneb, et õppeaine AA on ise esimeseks kokkupuuteks rekursiooniga Kepleri (251.−300.) ja Helsinki (126.−150.) Ülikoolis. Üheksas ülikoolis (Konstanz, Gent, Ulm, Aalborg, RPI, Chalmers, California, Princeton, MIT) on enne AA õpet üks rekursiooni käsitlev aine, kolmes ülikoolis (Bergen, Kataloonia, Aalto) kaks ainet ja Marylandi Ülikoolis (44.) kolm õppeainet. Keskmiselt käsitlevad rekursiooni enne AAd seega 1,13 õppeainet (arvestades Kepleri ja Helsinki Ülikooli) ja 1,29 õppeainet (mitte arvestades Kepleri ja Helsinki Ülikooli).

6.3 Õpiväljundid

48st ainest defineerisid eelteadmised kuus ainet. Kõik antud kuus ainet olid kas ained „Algoritmid“, „Andmestruktuurid“ või AA, mida õpetati kas 2. või 4. semestril. Teiseks semestriks eeldas näiteks RPI, et tudeng on rekursiooni näinud ja et tal on sellest baasarusaam [37]. Helsinki Ülikool [38] ja MIT [39] eeldasid, et läbitud on alg- ja edasijõudnute programmeerimiskursused ning sissejuhatus ülikoolitaseme matemaatikasse. Marylandi Ülikool nõudis 4. semestri aineks „Algoritmid“ [41] programmeerimise baaskontseptsioonide (k.a. rekursiooni) teadmist. California Ülikool (11. paremuselt) eeldas, et 2. semestril ainesse „Andmestruktuurid“ jõudnud tudeng on kompetentne programmeerija st. tunneb end mugavalt OOPi, rekursiooni, järjendite ja puudega [42]. RPI (301.−400.) eeldas samuti, et tudeng on kompetentne programmeerija, kuid alles 4. semestri aineks „Sissejuhatus algoritmidesse“ [43].

48 ainest ei maininud enda õpiväljundites rekursiooni 29 õppeainet. Ülejäänud 19 ainet tõid kombineeritult välja 25 erinevat tehnilist oskust. Populaarseimateks õpiväljunditeks olid vastavalt: tudeng teab mõistet ‘rekursioon’, tudeng oskab lahendada (lihtsamaid) rekursiivseid funktsioone ja rekursiivse algoritmi keerukuse analüüs.

Kõige põhjalikumalt kirjeldas õpiväljundeid KPÜ (126.−150.). Põnev ja spetsiifiline näide on antud ülikooli aine „Programmeerimine II“ õpiväljundid [44], milles kirjutatakse, et tudeng oskab teha kindlaks, kas antud lihtsa rekursiivse algoritmi efektiivsust saab parandada ja oskab vajadusel kujundada efektiivsema rekursiivse algoritmi, mis lahendab sama probleemi. KPÜ on tähelepanuväärne näide ülikoolist, mille õpetavate oskuste areng on õpiväljundite põhjal läbipaistev, täpne ja läbimõeldud.

Semestrite ja ainete kaupa kirjeldati tehnilisi oskuseid järgnevalt:

1. semester: „Programmeerimine I“ [45]:
    - tudeng teab rekursiooni kui mõistet/ kontseptsiooni;
    - tudeng oskab lahendada (lihtsamaid) rekursiivseid funktsioone.
2. semester: „Programmeerimine II“ [44]:
    - tudeng oskab disainida iteratiivseid ja rekursiivseid algoritme, et lahendada otsingu− ja läbivaatamisprobleeme kuhjadel, järjekordadel ja järjenditel kasutades vastavalt vajadusele sobivaid andmetüüpe;
    - tudeng oskab disainida iteratiivseid ja rekursiivseid algoritme, et lahendada otsingu- ja läbivaatamisprobleeme binaar-, *n*-aar- jm. puude kohta;
    - tudeng oskab kirjeldada rekursiivse algoritmi disainimise põhisamme;
    - tudeng oskab põhjendada lihtsamate rekursiivsete algoritmide korrektsust;
    - tudeng oskab lihtsa rekursiivse algoritmi korral määratleda, kas leidub lihtne viis, kuidas seda algoritmi samaväärseks iteratiivseks algoritmiks muuta (ja seda muutust läbi viia);
    - tudeng oskab eristada, kas antud lihtsa rekursiivse algoritmi maksumus, mis töötab vektoritel, kuhjadel, järjekordadel või puudel, on lineaarse või ruutkeerukusega (eeldades, et maksumus on üks kahest);
    - tudeng oskab teha kindlaks, kas antud lihtsa rekursiivse algoritmi efektiivsust saab parandada ja vajadusel kujundada efektiivsem algoritm.
3. semester: AA [46]:
    - tudeng oskab algoritmi keerukust kirjeldada ja analüüsida kasutades rekurrentseid seoseid;
    - tudeng oskab raskendada rekursiooni põhiteoreemi.

Toodud näites pakutakse tudengile väga täpne ülevaade, mida ja millisel semestril ta omandama peaks. Kui TÜs tegeletakse tehnilisemalt rekursiooniga alles AA kursuses (ning vahepealset rekursiooniteadmisi kinnistavat ainet ei ole), siis KPÜ näitest selgub, et rekursiooniga tegeletakse suures osas just eelnevas kahes programmeerimiskursuses ning tehniliste oskusteni, nagu rekursiivse algoritmi keerukuse analüüs, jõutakse alles aines AA.

Kokkuvõtlikult tuleb välja, et suur hulk ülikoole ja nende õppeainete õpiväljundeid kas ei maini rekursiooni üldse või teevad seda väga põgusalt ja napisõnaliselt. Tihtipeale ei

mainitud ainet õpiväljundis, kuid mainiti aine kavas. Sagedasti mainiti sõna või mõistet ‘rekursioon’, aga ka rekursiivseid andmestruktuure, andmetüüpe ning algoritme (ja nendest viimase kirjutamist).

6.4 Õppematerjalid

Eesmärgiga pakkuda huvitunud õppijale ja õppejõule põhjalikku materjali, millele toetada rekursiooni metoodilise käsitluse ülesehitamist, uurib töö populaarseid õpperaamatuid, mida mitmed ülikoolid on enda tudengite õpetamiseks usaldanud. Seega toob peatükk välja vaid materjali, mida otsustas rakendada üle ühe ülikooli.

Sissejuhatavas aines „Arvutiteaduse alused“ kasutasid Konstanzi [47] ja Ulmi Ülikool [48] W. Küchlini ja A. Weberi raamatut „*Einführung in die Informatik: Objektorientiert mit Java*” (ISBN 3-540-20958-1). J.E. Hopcrofti, R. Motwani ja J.D. Ullmani raamatut „*Introduction to Automata Theory, Languages and Computation*” kasutasid Bremeni [49] ja Johannes Kepleri Linzi Ülikool aines „Tõenäosus ja keerukus“ [50] ning Konstanzi Ülikool [47] aines „Teoreetiline andmeteadus“.

Tähelepanu keskpunktiks on materjal, mida kasutati programmeerimise või AA õpetamisel kuna just selle abil saab pakkuda välja võimalikult tõhusaid, ent mitte liigäärmuslike soovitusi rahvusülikoolile.

Ottmanni ja Widmayeri õpikut „*Algorithmen und Datenstrukturen*” (ISBN 3-8274-1029-0) kasutab Konstanzi Ülikool [47] nii aineks AA kui ka kursuseks „Arvutiteaduse alused“. KPÜ kasutab aga ainetes „Programmeerimine II“ [51] ja AA [50] M. A. Weissi raamatut „*Data Structures and Problem Solving Using C++*” (ISBN 0321205006).

Õppeaine AA spetsiifiliselt kasutavad Konstanzi [47], Genti [52], KPÜ [46] Aalto [53] ja Ulmi [48] ülikoolid äärmiselt menukaimaks osutunud T.H. Cormen, C.E. Leiserson jt. raamatut „*Introduction to Algorithms*” (ISBN: 978-0262033848). Antud raamat on kohustuslik kirjandus veel Konstanzi Ülikooli aines „Arvutiteaduse alused“ [47]; ülikoolide RPI [43], Marylandi [54] ja MIT [39] kursustes „Algoritmid“ ja Johannes Kepleri Linzi Ülikooli õppeaines „Tõenäosus ja keerukus“ [50].

AA õpetamiseks kasutavad Konstanzi [47] ja Ulmi Ülikool [48] veel U. Schöningu õpikut „*Algorithmik*” (ISBN: 3827410924). Konstanzi [47], Bergen [55], Genti [52] ja Princetoni [56] ülikoolides on kohustuslikuks kirjanduseks R. Sedgewick’u ja K. Wayne’i „*Algorithms*” (4. väljaanne) (ISBN: 032157351X).

TÜ kohustuslike materjalidega kattuvusi ei olnud.

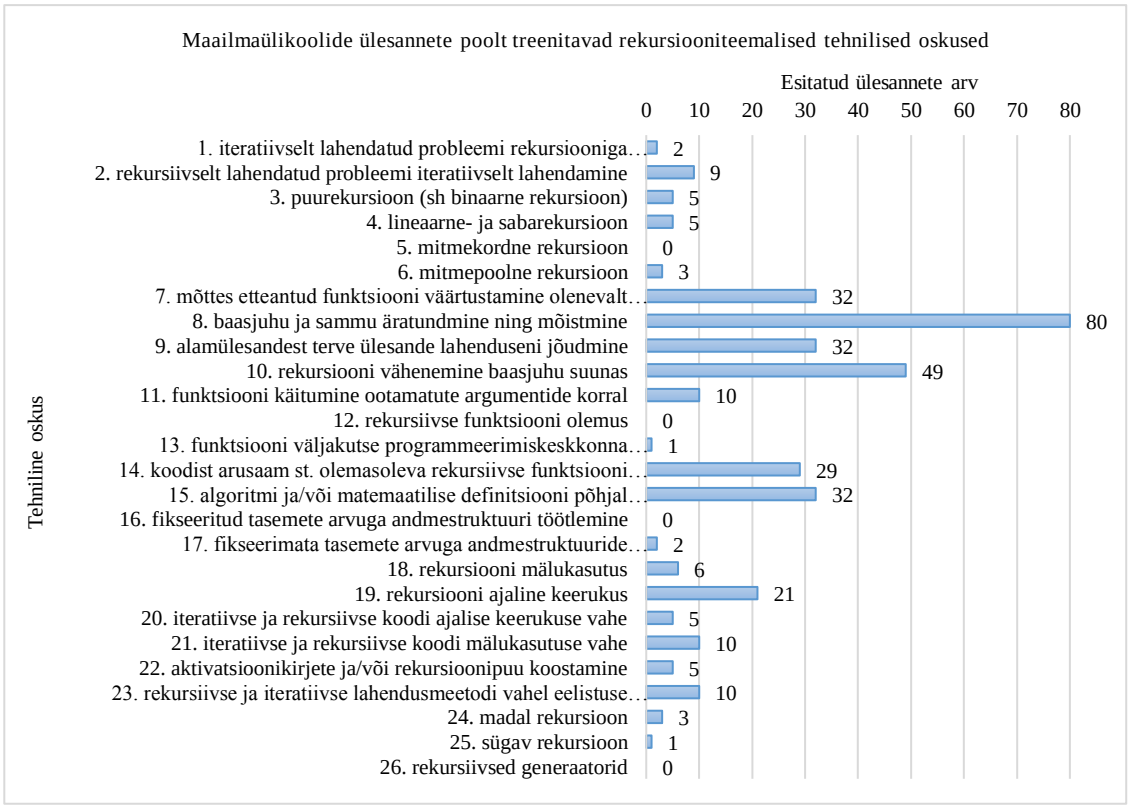
6.5 Tehnilised oskused läbi esitatud ülesannete

Tehnilised oskused

Uurime, kuidas välisülikoolid oma õpiväljunditele vastavad st. vaatleme esitatud ülesandeid ning seda, milliseid tehnilisi oskusi nende ülesannetega treenitakse.

Kõikide ülikoolide 48 ainest olid kättesaadavad 24 õppeaine ülesanded. Kokku esitasid need ained 137 ülesannet. Aritmeetilise keskmise järgi esitas iga aine 5,7 ja mediaani järgi 3,5 ülesannet.

Kõige vähem kättesaadavaid ülesandeid (1) olid viiel ainel: RPI „Sissejuhatus algoritmidesse“ kursusel, Helsinki Ülikooli AA ainel, California Ülikooli „Andmestruktuurid“ kahel kursusel ja Princetoni Ülikooli FP ainel. Enim ülesandeid (23) pakkus Princetoni Ülikooli (8. paremuselt) kursus „Informaatika: interdistsiplinaarne käsitlus“.



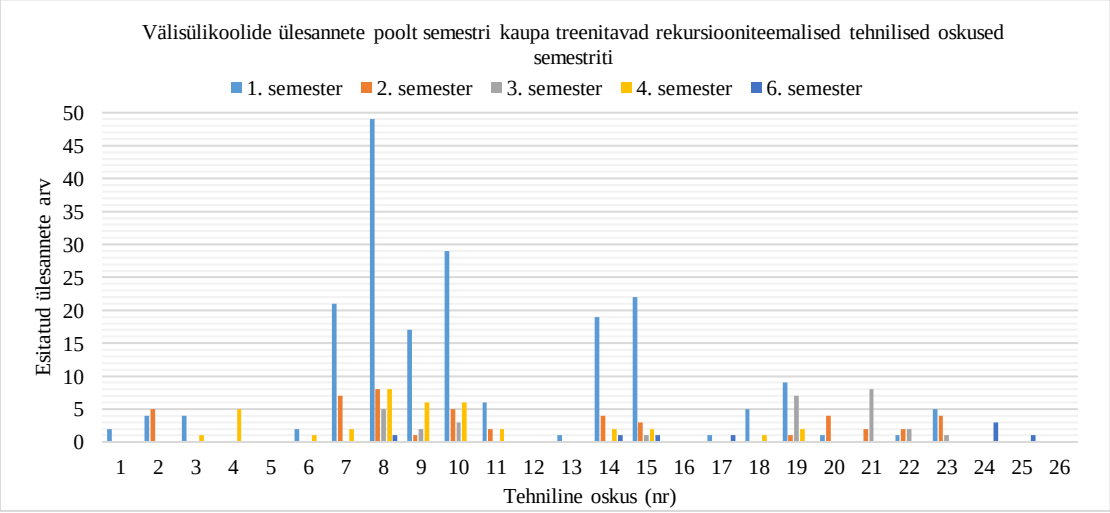
Joonis 5. Maailma ülikoolide ülesannete poolt treenitavad rekursiooniteemalised tehnilised oskused.

Joonis 5 visualiseerib, et välisülikoolid peavad kumulatiivse ülesannete arvu põhjal olulisimaks järgmist kuute tehnilist oskust:

1. (8) baasjuhu ja sammu äratundmine ning mõistmine;
2. (10) rekursiooni vähenemine baasjuhu suunas;
3. (7) mõttes etteantud funktsiooni väärtustamine olenevalt sisendist;
4. (9) alamülesandest terve ülesande lahenduseni jõudmine;
5. (15) algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek;
6. (14) koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine.

Tehnilistest oskustest esimene olulisim kattub TÜ olulisimaga. Teisalt, kui TÜ olulisuse kuuikus väärtustati oskuseid nagu (3) puurekursioon (sh. binaarne rekursioon), (11) funktsiooni käitumine ootamatute argumentide korral ja (22) aktiveerimiskirjete ja/või rekursioonipuu koostamine, siis välisülikoolid keskenduvad (7) mõttes etteantud funktsiooni väärtustamisele, (9) alamülesandest terve ülesande lahenduseni jõudmisele ja (15) algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanekule.

Võrdluse eesmärgil TÜga vaatleme semestrite kaupa, milliseid oskusi olulisemateks peetakse.



Joonis 6: Maailma ülikoolide ülesannete poolt semestri kaupa treenitavad rekursiooniteemalised tehnilised oskused.

Joonis 6 ei too välja 5. semestri ülesandeid kuna juhuse kombel ei esitanud valimisse sattunud ülikoolid sel semestril (kättesaadava põhjal) ühtegi ülesannet.

Keskendudes semestripõhiselt oskustele, illustreerib joonis 6, et I semestril (populaarseimaks õppeaineks „Arvuteaduse alused“, vt. tabel 2) keskenduti enim

oskustele (8) baasjuhu ja sammu äratundmine ning mõistmine, (10) rekursiooni vähenemine baasjuhu suunas ja (15) algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek. Esimesed kaks neist kattuvad populaarsete oskustega TÜ I semestril.

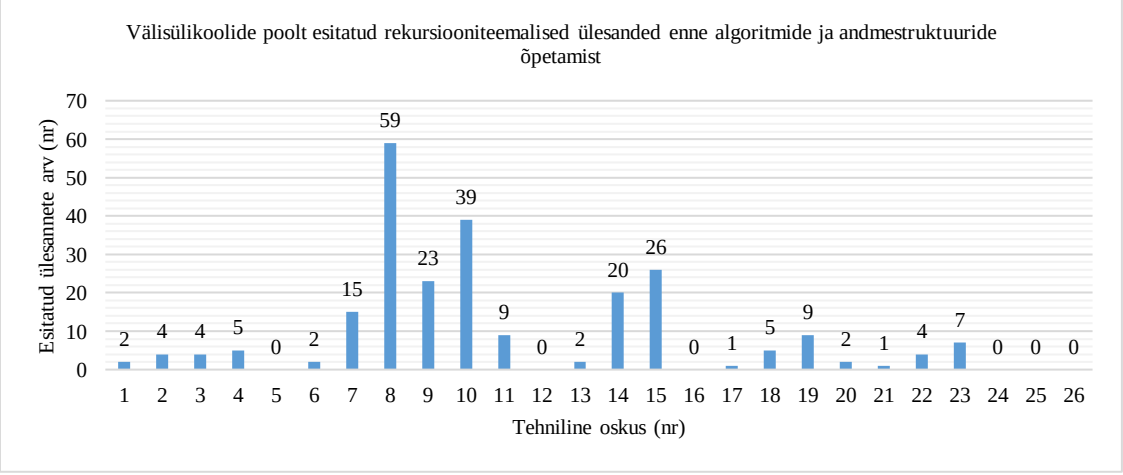
II semestri (populaarseimaks õppeaineks AA, vt. tabel 2) esikaksik kattus eelmise semestriga, ent (15) asemel väärtustati oskust (2) st. rekursiivselt lahendatud probleemi iteratiivselt lahendamist.

III semestril (populaarseimaks õppeaineks samuti AA, vt. tabel 2) kattus eelmiste semestritega võrreldes baasoskus (8), mis on 3. semestriks populaarsuselt kolmas, ent enim hinnati sealhulgas kaht uut tehnilist oskust – (21) iteratiivse ja rekursiivse koodi mälukasutuse vahe ja (19) rekursiooni ajaline keerukus.

Huvitaval kombel kattus IV semestri esimene (8) ja kolmas (10) olulisim 1. ja 2. semestri esikaksikutega. Populaarsuselt teisena väärtustati IV semestril oskust (9) alamülesandest terve ülesande lahenduseni jõudmine.

VI semestril (populaarseimateks õppeaineteks „Programmeerimiskeeled“ ja „Modelleerimine ja verifitseerimine“, vt. tabel 2) esitati ülesandeid vähem. Nende põhjal osutus populaarseimaks tehniline oskus (25) madal rekursioon. Ülejäänud ülesandeid esitati võrdselt üks kord.

Uurime kogutud andmete põhjal ka seda, milliseid ülesandeid ja tehnilisi oskusi peetakse maailma ülikoolides oluliseks enne ainet AA.



Joonis 7: Välisülikoolide poolt esitatud rekursiooniteemalised ülesanded enne AAd.

Joonis 7 põhjal selgub, et enne AA õpetamist on olulisimad järgmised kuus tehnilist oskust:

1. (8) baasjuhu ja sammu äratundmine ning mõistmine;
2. (10) rekursiooni vähenemine baasjuhu suunas;
3. (15) algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek;
4. (9) alamülesandest terve ülesande lahenduseni jõudmine;
5. (14) koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine;
6. (7) mõttes etteantud funktsiooni väärtustamine olenevalt sisendist.

TÜs on enne AA õpet vaid aine „Programmerimine“, mille olulisuse esikuuik kajastab, et niivõrd hinnaliseks ei peeta oskusi (15), (9), (14) ja (7). Nende asemel keskendub TÜ õppeaine LTAT.03.001 oskustele (3), (17), (1) ja (11).

Esitatud ülesanded

Uurides, milliseid ülesandeid ülikoolid esitavad ja milliseid tehnilisi oskusi nende abil tudengites enim arendada üritatakse, keskendus töö esimesele uurimisküsimusele. Töö vaatleb nüüd teist küsimust: milliseid ülesandeid, mis annavad sõnaselgelt märku, et probleem nõuab rekursiooni, esitasid ülikoolid üle maailma?

Lihtsamaid, algtaseme ülesandeid päriselunäidete põhjal toovad sõnaselgelt välja Princetoni, RPI ja Bergeni Ülikoolid. Näiteks toob Princetoni Ülikool [57] välja failisüsteemi töötlemise, kus on kaustad, mille sees on omakorda kaustad. Seda ülesannet põhjendatakse sellega, et taoline andmestruktuur on loomult iseendale viitav. Sama põhjendusega toob ülikool välja ülesandeid fraktaalidest ja funktsioonidest, mis rakendavad jaga-ja-valitse algoritme. Samuti mainib Princetoni Ülikool [57] laialt kasutatud Fibonacci jada (sh. kuldlõike ja jäneste reprodutseerimise ülesanne) näite ja ülesandena. Olgugi, et tegu on rekursiivse ülesandega, siis rõhutab ülikool põhjalikult, et halva Fibonacci algoritmi kirjapanekul võivad üleliigsed arvutused tekitada eksponentsiaalselt jääke (nt. arvutades 60ndat Fibonacci arvu arvutatakse 0ndat arvu miljardeid kordi), mis muudavad tehnoloogia arengut tühiseks [57].

Kõik kirjeldatud näited ja ülesanded kajastuvad ka mitmete teiste maailma ülikoolide esitatud ülesannete hulgas. Harva oli aga sõnaselgelt põhjendatud miks ja just rekursiooni kasutamine on otstarbekas.

Ülesandeid, mis nõuavad rekursiooni algteadmisi, tuuakse välja järgnevaid:

1. Listi tasandamise probleem, mille korral on mitterekursiivset funktsiooni keerulisem kirjutada [58].
2. Põimemeetodit nõudvad probleemid. Põimemeetod on fundamentaalselt rekursiivne ning selle rekursiivse sammu põhieesmärk on põimimisprotsess [58]. Teisalt mainib Bergeni Ülikool loengus [59], et põimemeetodi probleeme saab lahendada ka iteratiivselt.
3. Kiirsorteerimist nõudvad probleemid. Erinevalt põimemeetodist toob Bergeni Ülikool loengus [59] välja, et kiirsorteerimist ei saa iteratiivselt lahendada.
4. Ülesanded, mis kasutavad rekursiivseid andmestruktuure nagu järjendeid, puid ja graafe. Need andmestruktuurid võivad olla „juhuslikult väga suured" [60]. Kuna tegu on rekursiivsete andmestruktuuridega, siis vajame rekursiivseid funktsioone, et neilt andmeid kätte saada [60]. California Ülikool [64] toob välja, et kui mingi klassi objektil on selle sama klassi atribuudi väärtus, siis tegu on rekursiivse objektiga.
5. Lihtahelat rakendavad probleemid [61, 62, 64]. Lihtahel on definitsioonilt rekursiivne: „lihtahel on kas tühi või sisaldab tippu, milles on väärtus ja viit teisele lihtahelale" [61].
6. Puud rakendavad probleemid. Sümbolaritmeetilised avaldised on näide „väga harilikust puu andmestruktuurist" [62]. Sümbolaritmeetiline avaldis on rekursiivselt defineeritud andmetüüp, mille efektiivseks kasutuseks probleemi lahenduses on mõistlik kasutada rekursiooni [62].
7. Täieliku otsinguga probleemid ehk ülesanded, kus sammhaaval võetakse arvesse kõiki võimalusi lahenduseni jõudmiseks. Aalto Ülikooli sõnul [63] on rekursioon otsinguprobleemide jaoks loomulik valik. Konkreetne näide otsinguprobleemist on alamhulga summa leidmine. Probleemi matemaatiline sõnastus on järgnev [63]:

   > Kui meil on sisendhulk $W = \{w_1, w_2, \dots, w_n\}$ suurusega n (täisarv) ning täisarv *t*, siis kas leidub alamhulk $S \subseteq W$ nii, et $\sum_{z \epsilon S} z = t$?

   Olgugi, et rekursiivne lahendusmeetod ei ole kõige efektiivsem, siis paremat algoritmi ei ole välja mõeldud.

Vahva ülesandena esitab Bergeni Ülikool Microsofti intervjuuküsimuse – „kuidas sa selgitaksid rekursiooni 6-aastasele" [65]? Vastuseks on selgitada rekursiooni kellelegi,

kes on üks aasta noorem kui Sina ning paluda neil omakorda selgitada rekursiooni kellelegi, kes on üks aasta neist noorem. Jätkata senikaua, kuni leidub 7-aastane, kes seletab rekursiooni 6-aastasele.

Aalto Ülikool mainis kahte ülesannet, millal rekursiooni ei peaks kasutama. Esimene neist on Fibonacci (olgugi, et ülikool toob välja ka efektiivsema versiooni) ja teine binaarotsing, mida „saab (ja peaks) rakendama lihtsalt ilma rekursioonita" [65].

6.6 Õpetamismeetodid

Töö kolmandaks põhiküsimuseks on uurimine, milliste meetoditega õpetatakse tudengeid iseseisvalt rekursiooni kasutama ja aru saama, et probleemi lahendus nõuab rekursiooni.

Arukas lähenemisviis on enne õpetamise alustamist kõigepealt õpetatavate teadmustasemega tutvuda. California Ülikooli küsib 2. semestri alguses loenguslaididel tudengitelt (anonüümselt) tagasisidet, kuidas nad enda rekursiooni taset hindavad peale eelmist kursust, mis rekursiooni käsitles [67]. Tudengitel palutakse enda mugavust rekursiooniga hinnata skaalal „väga mugav", „mugav", „mõnevõrra mugav" ja „ei ole kunagi rekursiooniga kokku puutunud". Olgugi, et nimetatud skaalat annaks laiendada, siis on tegu õpetamismeetodiga, mida tarvitseks rakendada ka semestri keskel ja lõpus, et hinnata veelgi strateegilisemalt õppemeetodite toimivust.

Bergeni Ülikool kordab loengute alguses põgusalt üle, millega eelmine kord tegeleti. Chalmersi Ülikool teeb sama, ent loengu lõpus [68]. Käies üle teemad, mida loengus õpiti, saab tudeng enda õpitut kriitilise pilguga võrrelda ning vajadusel materjali üle korrata. Põnev loenguteemaline õppemeetod on muu hulgas see, et California Ülikool jaotab iganädalase ühe loengu hoopis kolmeks 50-minutiliseks loenguks [69]. Kõnealune töötab paremini, kui ainult ühe rekursiooniteemalise loengu pidamine, sest tudengid jõuavad vahepeal teemat iseendale lahti mõtestada ning seeläbi teemakohasemaid küsimusi küsida.

KPÜ korraldab nii tavalisi praktikume kui ka kord nädalas toimuvaid spetsiifilisemaid probleemilahendamise praktikume [46]. Bremeni Ülikool käsitleb aine praktikumis grupitööna vasakule kaldu kuhja ja Huffmani puud [70]. Vähemalt sama oluline on Princetoni Ülikooli meetod viia läbi grupiarutlus rekursiooniteemaliste küsimustega nagu näiteks „mida see tähendab kui öeldakse, et programm kasutab rekursiooni?" [71].

Helsinki Ülikoolis õpetatakse tudengeid aru saama, et probleem nõuab rekursiooni kõigepealt "mänguliste näidetega", nagu astendamine ja korrutamine, ja seejärel realistlikumaid rakendusi uurides [72]. Chalmersi Ülikool eelistab kohati enne definitsioone näiteid [68] tuua ning Aalto Ülikool esitab tudengile enne rekursiivse funktsiooni kirjutamist pseudokoodi, mida analüüsida [11]. Et tudeng oskaks mõista, millal millist rekursiivset lahendust eelistada toob Bergeni Ülikool loengus välja põime- ja kiirsordi erinevused [59] ning Aalto ülikool tabeli, mis illustreerib iteratiivsete ja rekursiivsete algoritmide erinevusi [11]. Chalmersi Ülikool vihjab materjalidele [73], mis soovitab mõelda juhust, kui rekursiivne samm ei kordu, et uurida, kas seda saab kasutada kui baasi. Samuti soovitab materjal mõelda, kas funktsiooni parameetreid saaks tükeldada ning millist osa tükeldusest saaks kasutada rekursiivses funktsioonikutses.

RPI toob välja juhud, kus rekursiooni ei peaks eelistama, nt. võrdleb faktoriaali rekursiivse lahendusmeetodi keerukust iteratiivse meetodiga [61].

Marylandi Ülikool märgib oma praktikumimaterjalides selgelt ära, millal peaks rekursiooni kasutama, aga esitab oma tudengitele loengumaterjalides ka malli sabarekursiivsete funktsioonide lahendamiseks. Väljapakutud malli rakendati faktoriaali näitega [74]:

```
let func x =
   let rec helper arc acc =
      if (base case) then acc
      else
         let arg' = (argument to recursive call)
         let acc' = (updates accumulator)
         helper arg' acc' in (*end on helper fun*)
   helper x (initial val of accumulator)
;;
```

```
let fact x =
   let rec helper arc acc =
      if arg = 0 then acc
      else
         let arg' = arg - 1 in
         let acc' = acc * arg in
         helper arg' acc' in (*end on
helper fun*)
   helper x 1
;;
```

Joonis 8: Ühe argumendiga sabarekursiivse funktsiooni mall [74].

Joonis 8 toimib kooskõlas tekstiga, mis selgitas tudengile sabarekursiooni olulisust. Rõhutatakse, et järjendiga töötades on ohtlik igal rekursiivsel funktsioonikutsel väljakutsekaadrit edastada. Mahukate sisendite korral (st. sügava rekursiooni korral) soovitati eelistada sabarekursiooni [74].

Sarnaselt toob California Ülikooli soovituslik kursus CS 61AS välja „kolm harilikku rekursioonimustrit“, mis võivad tudengit probleemilahendusel abistada [75]:

1. „Kõik“ muster st. kogume kokku kõik tulemused, mis tekivad iga sõna- või lauseelemendi millekski muuks muundamisel. Enamus „kõik“ mustrid koosnevad ühest baasist ja ühest sammust.

2. „Vali“ muster st. valime mõned elemendid ja filtreerime ülejäänud. Enamus „vali“ mustritel on üks baasjuht, aga kaks rekursiivset sammu. Peame ühes sammus otsustama, kas soovime tagastatud väärtuse esimest elementi hoida või mitte. Kui otsustame elementi hoida, siis hoiame elementi iseennast mitte funktsiooni elemendist.
3. „Kogu“ muster st. kombineerime iga argumendi elementide tulemused üheks tulemuseks. Kasutame nn. „liitjat“ (plussmärk või kindel sõne), et ühendada käesolev väärtus rekursiivsest funktsioonikutsest tagastatud väärtusega. Baasjuht testib, kas argument on tühi.

RPI pühendas ühe loengu probleemilahenduste õppimisele, mille raames toodi samuti välja kolm sammu (ideede genereerimine ja hindamine; ideede kajastamine koodis ja detailid), kuidas rekursiivseid algoritme disainida ja kirjutada [61:13].

Chalmersi Ülikool [76] ja MIT [77] kasutavad järjepidevalt testide kirjutamist ülesannete osana. Rekursiooni õpet toetavad omakorda e-materjalides illustreerivad animatsioonid nagu nt. Aalto Ülikooli e-raamatus [11] rakendust leidnud interaktiivsete küsimuste esitamisvormid, mis kohe tagasisidet pakkusid.

Sarnaselt aitavad rekursiivsete andmetüüpide mõistmisele kaasa ülikoolide viited veebilehekülgedele, mis andmestruktuure visualiseerivad. Välja saab tuua neli populaarsemat, mida kasutasid oma materjalides Kepleri [51], Chalmersi [78] ja Bergeni [65:15] ülikoolid:

1. San Fransisco Ülikooli materjal andmestruktuuride ja algoritmide visualiseerimiseks: https://www.cs.usfca.edu/~galles/visualization/Algorithms.html;
2. http://sorting.at/;
3. https://imgur.com/gallery/GD5gi;
4. https://visualgo.net/en/list.

Rekursiooniga seonduvaid mõisteid on palju. On hea, kui tehnilised oskused on ära jaotatud ainete vahel, sest kõige meelde jätmine ühe aine põhjal ei ole ratsionaalne eeldus. Sellise olukorra puhul on RPI e-õppematerjal heaks näiteks materjalist, mis tegi uue teema alguses ja lõpus kokkuvõtte põhilisest, mida mõistma pidi [58].

California Ülikool selgitas õppekava I semestri aines, et kui tudeng peab uurima mitut võimalust samaaegselt, siis tasuks rusikareeglina kaaluda puurekursiooni rakendamist [79]. Samuti soovitati rekursiooni kirjutamist õppides panna esmalt kirja baasjuhud [80].

Rekursiooni õppimisele aitab veel kaasa nt. Aalto Ülikooli e-raamatu erialasõnastik viidetega aine e-raamatule: https://plus.cs.hut.fi/o1/2018/wNN/glossary/ ja see, et California Ülikool on kogunud kokku tudengite ja õppeassistentide koostatud materjale [69]. Mitu perspektiivi ja probleemile lähenemise viisi on igati positiivne.

Olulisena toodi mitmeid kordi välja oma koodi usaldamist (ingl *recursive leap of faith*) st. uskumine, et rekursioon teeb seda, mida soovime.

Käesolev peatükk kirjeldas enim meetodeid, mille abil teha õpinguid põnevamaks ja interaktiivsemaks. Siiski leidus peatükis ka väljapaistvaid ja efektiivseid meetodeid, nagu rekursiivse funktsiooni mall, rusikareeglina puurekursiooni kaalumine ja grupitöö, mille näitel saab TÜ rekursiooni käsitlust veelgi parendada.

6.7 Järeldused

Tugevused

Välisülikoolid tõid välja oodatust rohkem kohustuslikku kirjandust. Kokku esitasid 13 õppeainet 55 kohustuslikku õppematerjali. Üldise positiivse näitena paistis silma KPÜ nii põhjaliku kirjandusloetelu kui ka õpiväljunditega.

Positiivne oli näha, et TÜs populaarseimana käsitletav tehniline oskus (8) on populaarseim ka välisülikoolides. Populaarsuse statistika põhjal läksid üleüldiselt tehnilised oskused semestriti keerulisemaks, ent ka kattusid. Nimetatu annab märku sujuvast üleminekust semestrite ja ainete vahel.

Eeskujuks Eesti ülikoolidele tõid vähemalt mõned ülikoolid sõnaselgelt välja ülesandeid, mis rekursiooni nõuavad. Ülesandeid pakuti nii algajatele kui ka õppuritele, kellelt algteadmisi oodatakse. Veelgi enam toodi välja rohkelt meetodeid, kuidas rekursiooni õpetamist põnevamaks ja interaktiivsemaks teha. Silmapaistva meetodina mainis California Ülikool [79] I semestri aines, et kui tudeng peab uurima mitut võimalust samaaegselt, siis tasuks rusikareeglina kaaluda puurekursiooni rakendamist.

Kitsaskohad

Olenemata mahukast valimist leidus üllatavalt vähe põhjalike ja avatud materjalidega ülikoole. Töös vaadeldud 25 ülikooli seas ei leidunud ühtegi avatud materjalidega ülikooli, kelle paremushinnang *Times Higher Education* portaalis oleks olnud sama rahvusülikooliga (251.-300.).

Sarnaselt TÜle selgus, et enne AA õpet on keskmiselt üks õppeaine. Olgugi, et tasemed, mida maailmas tudengitelt AAs oodatakse, varieerusid märgatavalt, siis ei näi vaid üks aine piisav. Samuti selgus, et näidete arv kaalumas üle harjutuste ja küsimuste arvu ei ole probleem vaid TÜs. Loengumaterjalidesse küsimuste põimimine parandab dialoogi õppuritega ning aitab neil paremini materjali mõtestada.

Eelteadmiste nõudmisi (või nende puudumist) defineeris 6 õppeainet vaadeldud 48st. Valimis ei maininud 29 (60,4%) ainet oma õpiväljundites rekursiooni. Suur hulk ülikoole ja nende õppeainete õpiväljundid kas ei maini rekursiooni üldse või teevad seda väga põgusalt ja napisõnaliselt. Olgugi, et rohkem, kui Eesti ülikoolides, siis oli ka maailma ülikoolide hulgas harva sõnaselgelt põhjendatud ülesannete juures miks ja millal just rekursiooni rakendamine on otstarbekas.

Kokkuvõte

Aineid, mis rekursiooni käsitlevad, on põhjusega palju. Rekursiooni õpe toimub uuritu põhjal ideaalis tükiti ja progresseeruvalt läbi mitme õppeaine ning semestri. Maailma ülikoolide näitel õpetatakse rekursiooni esimesest kuuenda semestrini. Selle asemel, et TÜ viimaset Informaatika õppekavast aine „Programmeerimiskeeled“ valikaineks muudeti, võiks rekursiooni hoopis mõõdukalt rohkemates kursustes käsitleda. Põhjuseks on see, et rekursioon ilmutab oma täispotentsiaali just enim funktsionaalsetes programmeerimiskeeltes, mis oli ka maailma ülikoolides üheks populaarseimaks õppeaineks.

Kogutud andmete põhjal saab järeldada, et valimkõrgkoolide esinelik, Marylandi Ülikool, California Ülikool, Princetoni Ülikool ja MIT esitasid enim töös kasutatud ülesandeid ja õppemeetodeid. Olgugi, et kohati olid nende esitatud ülesanded (nt. fraktaalid) keerukamad ja ulatuslikumad, siis ei unustanud ülikoolid algmõistete põhjalikku käsitlust. TÜ esitas samuti rohkelt ülesandeid ning on võrdluses maailma ülikoolidega eeskujulik ja põhjalik. Parendusi ja mõtteid leidus enamustes ülikoolidest, olenemata nende positsioonist *Times Higher Education* paremusjärjestuses.

Tendents, et õpiväljundites kirjeldatakse rekursiooni hõredalt, jätkus. Siiski sai välisülikoolide sünteesist teha mitmesuguseid järeldusi, nagu olulisimad oskused, mida enne AAd õpetada, ning omandada rohkelt mõtteid, kuidas rekursiooni vajalikkust tudengitele paremini õpetada.

Populaarseimateks õpiväljunditeks olid vastavalt: tudeng teab mõistet ‘rekursioon’, tudeng oskab lahendada (lihtsamaid) rekursiivseid funktsioone ja rekursiivse algoritmi keerukuse analüüs. Aine kirjeldustest mainiti aga sagedasti vaid sõna ‘rekursioon’, mõistet ’rekursioon’ ning rekursiivseid andmestruktuure, andmetüüpe ning algoritme (ja nendest viimase kirjutamist).

Kui TÜ tehniliste oskuste olulisuse kuuikus väärtustati oskuseid nagu puurekursioon (sh. binaarne rekursioon), funktsiooni käitumine ootamatute argumentide korral ja aktiveerimiskirjete ja/või rekursioonipuu koostamine, siis maailma ülikoolid keskendusid mõttes etteantud funktsiooni väärtustamisele, alamülesandest terve ülesande lahenduseni jõudmisele ja algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanekule. Huvitaval kombel ei pööratud ühegi ülikooli ülesannetes sõnaselgelt palju rõhku tehnilistele oskustele (24) madal rekursioon, (25) sügav rekursioon ja (26) rekursiivsed generaatorid.

Samamoodi nägime, et kui välisülikoolid pidasid enne AA õpet oluliseks oskusi (8), (10), (15), (9), (14) ja (7), siis TÜs on enne AA õpet vaid aine „Programmerimine“, mille olulisuse esikuuik kajastab, et oskusi (15), (9), (14) ja (7) niivõrd hinnaliseks ei peeta. Nende asemel keskendub õppeaine LTAT.03.001 oskustele (3), (17), (1) ja (11).

TÜ kohustuslike materjalidega kattuvusi ei olnud.

Algtaseme ülesannetest osutusid populaarseimateks failisüsteemi töötlemine, Fibonacci jada efektiivsus ning fraktaalid. Elementaartõdesid nõudvateks harjutusteks olid probleemid, mille lahendus hõlmas rekursiivseid andmetüüpe nagu lihtahelat, puud või graafi. Samuti toodi välja, et rekursiooni tarvilikkusele viitavad kiirmeetodi, põimemeetodi ja täieliku otsinguga ülesanded.

Õppemeetodite läbivaks jooneks olid e-õpikute ja/ või e-raamatute kasutamine. Materjalides tehti pidevalt kokkuvõtteid ning küsiti entusiastlikult tagasisidet. Üks kõrgkool eelistas lühemaid, ent sagedasemaid loenguid. Samuti viidi kohati läbi spetsiifilist probleemilahenduse praktikumi ja loengut. Et tudengid end rekursiooniga

mugavamalt tunneksid, võrreldi nt. rekursiooni ja iteratsiooni, toodi välja rekursioonimustreid ning soovitati enda koodi toimimist usaldada.

Saab järeldada, et konkreetset ühte ja õiget õpetamis- või käsitlusmeetodit ei ole. Tudengites huvi äratamiseks on esmalt mõistlik esitada neile päriselu ülesandeid. Kui rekursiooni kontseptsioon kinnistub, siis tasub liikuda edasi piisava taktikalise harjutamise, lahtimõtestamise ja tagasisidega. Sealhulgas on oluline panna tudeng huvituma ka rekursiooniga kaasneva mõtlemisoskuse arengust.

7. Ülesannete komplekt

Ülesannete komplekti põhieesmärk on TÜ tudengitele rekursioonialaste teadmiste kinnistamise ja aineks AA parema valmisoleku võimaldamine.

Õppurite töö lihtsustamiseks on ülesannete juures välja toodud tehnilised oskused. Nii lähenedes saab tudeng enda nõrgemaid külgi iseseisvalt ära tunda ja arendada.

Ülesannete komplekt keskendub harjutustele, mis annavad märku rekursiooni vajalikkusest. Seeläbi saab tudeng suurendada võimalusi, et oskab rekursiooni rakendada hiljem iseseisvalt.

Komplekt treenib õppuritele vajaminevaid oskusi kuna on koostatud töö uurimusest selgunud teadmuse põhjal. Esmalt võtab komplekt arvesse TÜ käesolevaid õpiväljundeid, mida ainetega „Programmeerimine“ ja AA tudengites treenitakse. Teiseks võtab komplekt arvesse tehnilisi oskusi, mida TÜ, TalTech ja välisülikoolid ülesannete arvu põhjal enim harjutavad. Kolmandaks eelistab komplekt ülesandeid, mis treenivad oskusi, mida on tarvis enne AA õppimist. Seega üritab komplekt võtta arvesse lihtsamaid ülesandeid, millega alustada rekursiooni õpet, aga ka ülesandeid, mis aitavad AAd edukamalt sooritada. Samuti on tugevalt eelistatud ülesanded, mis vihjavad rekursiooni kasutamise vajadusele ning iteratsiooni ja rekursiooni vahel eelistuse tegemisele. Viimasena on komplektis ülesandeid, mis nõuavad rekursiooni kasutust iteratiivse lahenduse liigse keerukuse tõttu. Seeläbi treenib ülesannete komplekt tudengi võimet eelistada ühte lahendusmeetodit teisele.

Uurimusest selgus muu hulgas üldisi soovitusi rekursiooni efektiivsemaks õpetamiseks, mida on ülesannete komplekti üritatud võimalikult palju põimida. Esiteks on oluline õpiväljundite, ainete sisude ja ülesannete eesmärkide (ning treenitavate tehniliste oskuste) informatiivsus. Samuti on oluline, et õpiväljundid oleksid läbi ainete koordineeritud. Headeks toetavateks materjalideks on e-õpikud ja interaktiivsed õppematerjalid, millele perioodiliselt vihjata. Veelgi enam on tudengitele ülesannete juures lahendamisel abiks peidetud vihjed ja lahendused, millest viimased peaksid asuma teisel veebilehel, et vältida lihtsamat vastupanu. Kaasa aitavad ka ülesannete vormistamine erinevatele programmeerimiskeeltele, viide erialasõnastikule, abistavad küsimused hätta jäänutele, võimalus anda õppematerjalile tagasisidet ja kokkuvõte õpitust.

Viimasena on ääretult oluline selgitada tudengitele miks on rekursiooni õppimine üldse oluline. Näiteks toob Princetoni Ülikool FP õppimise olulisuse selgituseks välja, et FP on

seotud ja kasutusel andmekeskuste modernsetes ja massiivselt paralleelsetes rakendusprogrammides. Google kurikuulsal *map-reduce* masinal, mida kasutatakse nt. veebilehtede olulisuse järgi reastamiseks, on juured FP keeltes. [60]. Tuletades meelde töö 3. peatükis uuritut, on sagedase rekursiooni kasutuse tõttu FP tuntud. [11]

7.1 Väljapakutud ülesanded

Väljapakutud ülesannete komplekt on mõeldud e-õpikusse või veebilehele, kus ülesanded on õppuritele pidevalt kättesaadavad. Leht, mis kasutab ülesannete komplekti, peab komplekti lõpus pakkuma lugejale võimaluse anda tagasisidet kommenteerimisvormi, emailiaadressi või mõne muu lahenduse abil.

Komplektis toodud vihjed ja vastused peaksid olema peidetud. Eelistatult asuksid vastused teisel veebiaadressil.

Võimalusel tasuks komplekti käsitleda eraldi (rekursiivse) probleemilahendamise praktikumis. Täispikk ülesannete komplekt on leitav Lisas II.

8. Kokkuvõte

Käesolev bakalaureusetöö astub väikese sammu parema eestikeelse rekursioonialase kirjanduse suunas. Töö uuris, analüüsis ja võrdles Tartu Ülikooli, Tallinna Tehnikaülikooli ja välisülikoolide rekursiooni käsitlust. Uurimuse tulemina selgunud ülesannete, tehniliste oskuste, õppemeetodite ja –materjalide rohkus rõhutas, et rekursiooni on võimalik tõhusamalt ja koordineeritumalt õpetada.

Töö üheks saavutuseks oli uurimuse käigus iga ülikooligrupi raames kolmele uurimusküsimusele vastamine.

Esiteks selgus, et suur hulk tudengitele esitatud ülesannetest treenib rekursiooni algteadmisi st. baasjuhu ja sammu äratundmist ning sügavuti mõistmist. Kui Tartu Ülikooli harjutused keskendusid enamasti puurekursioonile, funktsiooni käitumisele ootamatute argumentide korral ja aktiveerimiskirjete ja/ või rekursioonipuu koostamisele, siis välisülikoolid pidasid olulisemaks mõttes funktsiooni väärtustamist, alamülesandest tervikliku lahenduseni jõudmist ning algoritmi ja/ või matemaatilise definitsiooni põhjal funktsiooni kirjapanekut.

Võrdluses ilmnes, et Tartu Ülikool esitas vähevõitu ülesandeid, mis andsid sõnaselgelt märku, et probleem nõuab just rekursiooni. Maailma ja Eesti ülikoolid tõid välja algtasemele sobilikke ülesandeid, nagu failisüsteemi töötlemine, Fibonacci jada (efektiivsus) ning fraktaalid. Edasijõudnutele anti märku, et probleem nõuab rekursiooni rekursiivsete andmetüüpide, nagu lihtahela, puu ja graafi korral. Lisaks tõid kõrgkoolid ülesannete raames välja sabarekursiooni, kiirmeetodi, põimemeetodi ja täieliku otsingu. Kontrastina vihjas Tartu Ülikool, et rekursioon ei sobi ülesannete jaoks, mis nõuavad millegi lõputut kordust.

Kolmandana selgus, et populaarseteks meetoditeks, mis õpetavad tudengeid iseseisvalt rekursiooni tarvilikkusest aru saama, on ülesanded interaktiivsetes e-õpikutes, eraldi probleemilahenduse praktikumid ja/ või loengud ning päriselulised näited. Samuti toodi välja funktsiooni rekursioonipuu koostamist, iteratiivse ning rekursiivse lahenduse sooritusvõrdlusi ja rekursioonimustreid. Veel rõhutati tudengitele rekursiooni kasutamise vajadust iteratiivselt lahendamiseks mõeldud ülesandega, mille rekursiivne lahendus oli lihtsam. Viimasena oli populaarseks soovituseks usaldada, et rekursiivne kood teeb seda, mida soovime.

Osutus, et konkreetset ühte ja õiget õpetamis– või käsitlusmeetodit ei ole. Tudengites huvi äratamiseks on mõistlik esmalt esitada päriseluga seonduvaid näiteid ning hiljem jätkata taktikalise harjutamise, ülesannete lahtimõtestamise ja pideva tagasiside küsimisega. Samuti on väärtuslik panna tudeng huvituma rekursiooniga kaasnevast mõtlemisoskuse arengust. Tartu Ülikooli rekursiooni käsitluse parendamisel on oluliseks põhjalikumad ning täpsemad õpiväljundid, ülesannete esitamisel treenitavate oskuste mainimine, materjalide interaktiivsus ning pidev tagasiside küsimine.

Töö autor panustas metoodiliste käsitlusviiside ülesehitamisse, kirjanduse arengusse ja tulevaste tudengite paremasse valmisolekusse kursuses „Algoritmid ja andmestruktuurid“ suuresti ka töö tulemi – ülesannete komplekti – abil. Koostatud komplekt võimaldab õpetada tudengitele rekursiooni suurema kindlusega.

Ülesannete komplekti tasuks edaspidi katsetada avatud materjalina õppurite peal. Tagasisidest komplektile võib omakorda areneda välja veel põnevat eestikeelset rekursioonialast kirjandust.

Viidatud kirjandus

[1] B. Harvey, M. Wright ja H. Abelson, *Simply scheme: Introducing Computer Science*, 2. väljaanne. Cambridge: The MIT Press, 1999, lk 242.

[2] J. Tessler, B. Beth ja C. Lin, "Using cargo-bot to provide contextualized learning of recursion", *Proceedings of the ninth annual international ACM conference on International computing education research - ICER '13*, 2013, lk 2.

[3] P. Brodanac, "Recursions and how to teach them", *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2017, lk 740–745.

[4] B. Stephenson, "Using graphical examples to motivate the study of recursion", *Journal of Computing Sciences in Colleges*, vol. 25 no. 1, 2009, lk 42–50.

[5] O. Hazzan, T. Lapidot and N. Ragonis, *Guide to Teaching Computer Science: An Activity-Based Approach*. New York, NY: Springer, 2015 lk 194.

[6] C. George, "EROSI - Visualising recursion and discovering new errors", *Proceedings of the thirty-first SIGCSE technical symposium on Computer science education - SIGCSE '00*, vol. 32, no. 1, 2000, lk 305–309.

[7] M. Rubio-Sánchez, J. Urquiza-Fuentes ja C. Pareja-Flores, "A gentle introduction to mutual recursion", *ACM SIGCSE Bulletin*, vol. 40, no. 3, 2008, lk 235–239.

[8] M. Morazán, "Functional video games in CS1 II". *Trends in functional programming, 12th international symposium, TFP 2011, Madrid, Spain, May 16–18, 2011, Revised selected papers*, vol. 7193. Berlin, Germany: Springer; 2011, lk 146–162.

[9] Tartu Ülikooli Arvutiteaduse Instituudi kodulehekülg. https://www.cs.ut.ee/et/oppimine/bakalaureuseope (01.05.2019)

[10] Y. Liang, *Introduction to Java Programming*, 10. väljaanne. Boston: Pearson, 2015, lk 726.

[11] Aalto Ülikooli e-õpik. https://plus.cs.hut.fi/o1/2018/w11/ch02/ (01.05.2019)

[12] Middlesexi Ülikooli kodulehekülg. https://www.mdx.ac.uk/about-us/our-people/staff-directory/profile/george-carlisle (01.05.2019)

[13] Tartu Ülikooli õppeaine LTAT.03.001 kursuse "Programmeerimine" e-õpik. https://progeopik.cs.ut.ee/11_rekursioon.html (01.05.2019)

[14] Tartu Ülikooli õppeaine LTAT.03.001 "Programmeerimine" kursuse kodulehekülg. https://courses.cs.ut.ee/2018/programmeerimine/fall/ (01.05.2019)

[15] Tartu Ülikooli õppeaine LTAT.03.005 "Algoritmid ja andmestruktuurid" ainekava info. https://www.is.ut.ee/rwservlet?oa_ainekava_info.rdf+1348331+PDF+0+application/pdf (01.05.2019)

[16] Tartu Ülikooli õppeaine LTAT.03.006 "Automaadid, keeled ja translaatorid" ainekava info. https://www.is.ut.ee/rwservlet?oa_ainekava_info.rdf+1349795+PDF+0+application/pdf (01.05.2019)

[17] Tartu Ülikooli õppeaine LTAT.03.006 „Automaadid, keeled ja translaatorid“ kursuse kodulehekülg. https://courses.cs.ut.ee/2018/AKT/spring (01.05.2019)

[18] Ahti Peder, Härmel Nestra. Algoritmid ja andmestruktuurid. Tahvlipraktikum. 2018. https://moodle.ut.ee/pluginfile.php/989588/mod_resource/content/3/peafail_27082018.pdf (02.05.2019)

[19] J. Zelle, *Python Programming: An Introduction to Computer Science*, 3. väljaanne. Portland: Tom Sumner, 2016, lk 498-501.

[20] ÕIS II: LTAT.03.001 „Programmeerimine“ õppijate tagasiside 17/18 S PÕ LT ja 18/19 S PÕ LT. https://ois2.ut.ee/#/courses/LTAT.03.001/version/lt-2019-autumn-fulltime-c6/feedback. (02.05.2019)

[21] ÕIS II: LTAT.03.005 „Algoritmid ja andmestruktuurid“ õppijate tagasiside 17/18 ja 18/19. https://ois2.ut.ee/#/courses/LTAT.03.005/version/lt-2018-autumn-fulltime-c6/feedback (02.05.2019)

[22] Aalto Ülikool õppeaine CS-A1110 „Programming 1“ erialasõnastik. https://plus.cs.hut.fi/o1/2018/wNN/glossary/ (02.05.2019)

[23] Tartu Ülikooli õppeaine LTAT.03.005 „Algoritmid ja andmestruktuurid“ 5. praktikum. https://courses.cs.ut.ee/2019/AKT/Main/Praks5 (02.05.2019)

[24] Tartu Ülikooli õppeaine LTAT.03.005 „Algoritmid ja andmestruktuurid“ 4. loeng. https://moodle.ut.ee/pluginfile.php/14788/mod_resource/content/8/loeng4aasta2018.pdf (02.05.2019)

[25] Tartu Ülikooli õppeaine LTAT.03.005 „Algoritmid ja andmestruktuurid“ 3. loeng. https://moodle.ut.ee/pluginfile.php/12460/mod_resource/content/7/loeng3aasta2018.pdf (02.05.2019)

[26] IT-Kolledži kodulehekülg. https://www.itcollege.ee/sisseastujale/ (02.05.2019)

[27] Tallina Tehnikaülikooli Informaatika õppekava IAIB 17/19. https://ois.ttu.ee/portal/page?_pageid=37,674560&_dad=portal&_schema=PORTAL&p_action=view&p_fk_str_yksus_id=50001&p_kava_versioon_id=50512&p_net=internet&p_lang=ET&p_rezhiim=0&p_mode=1&p_from= (02.05.2019)

[28] Tallina Tehnikaülikooli õppeaine ITI0102 „Programmeerimise algkursus“ õpiväljundid. https://ois.ttu.ee/portal/page?_pageid=37,674581&_dad=portal&_schema=PORTAL&p_action=view&p_id=115200&p_session_id=72910354&p_public=1&p_mode=1&keel=ET (02.05.2019)

[29] Tallina Tehnikaülikooli õppeaine ITI0204 „Algoritmid ja andmestruktuurid“ õpiväljundid. https://ois.ttu.ee/portal/page?_pageid=37,674581&_dad=portal&_schema=PORTAL&p_action=view&p_id=115200&p_session_id=72910354&p_public=1&p_mode=1&keel=ET (02.05.2019)

[30] Tallina Tehnikaülikooli õppeaine ITI0102 "Programmeerimise algkursus"

Courses leht. https://courses.cs.ttu.ee/pages/ITI0102 (02.05.2019)

[31] Tallina Tehnikaülikooli õppeaine ITI0202 "Programmeerimise põhikursus" Courses leht. https://courses.cs.ttu.ee/pages/ITI0202 (02.05.2019)

[32] Tallina Tehnikaülikooli õppeaine ITI0102 "Programmeerimise algkursus" e-õpik. https://ained.ttu.ee/pydoc/recursion.html (02.05.2019)

[33] Tallina Tehnikaülikooli õppeaine ITI0102 "Programmeerimise algkursus" loenguslaidid. https://ained.ttu.ee/pydoc/lectures/loeng_recursion.pptx (02.05.2019)

[34] Tallina Tehnikaülikooli õppeaine ITI0202 "Programmeerimise põhikursus" e-õpik. https://ained.ttu.ee/javadoc/rekursioon.html (02.05.2019)

[35] *Times Higher Education* 2019. aasta maailmaülikoolide paremusjärjestus. https://www.timeshighereducation.com/world-university-rankings/2019/subject-ranking/computer-science#!/page/0/length/25/sort_by/rank/sort_order/asc/cols/stats (02.05.2019)

[36] Arvutiteaduse Instituudi ERASMUS partnerid. https://www.cs.ut.ee/sites/default/files/cs/erasmus_partnerid_4.pdf (02.05.2019)

[37] Rensselaeri Polütehnilise Instituudi õppeaine CSCI 1200 "Data Structures" õpiväljundid. https://www.cs.rpi.edu/academics/courses/spring17/ds/syllabus.php (02.05.2019)

[38] Helsinki Ülikooli õppeaine TKT20001 "Tietorankenteet ja algoritmit" õpiväljundid. https://courses.helsinki.fi/en/tkt20001 (02.05.2019)

[39] Massachusettsi Tehnoloogiainstituudi õppaine 6.006 "Introduction to Algorithms" aine kirjeldus. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/syllabus/ (02.05.2019)

[40] Genti Ülikooli 2018-2019 Informaatika õppekava. https://studiegids.ugent.be/2018/NL/FACULTY/C/BACH/CBINFO/CBINFO.html (02.05.2019)

[41] Marylandi Ülikool õppeaine CMSC 251/351 "Algorithms" õpiväljundid ja eeldused. http://www.cs.umd.edu/~samir/251/251.html (02.05.2019)

[42] California Ülikooli õppaine CS 61 B "Data Structures" e-raamat. https://joshhug.gitbooks.io/hug61b/content/ (02.05.2019)

[43] Rensselaeri Polütehnilise Instituudi õppeaine CSCI 2300 "Introduction to Algorithms" aine kirjeldus. https://www.cs.rpi.edu/~eanshel/2300/2300syllabus.html (02.05.2019)

[44] Kataloonia Polütehnilise Ülikooli õppaine PRO2 "Programming II" õpiväljundid. https://www.upc.edu/content/grau/guiadocent/pdf/ing/270005 (02.05.2019)

[45] Kataloonia Polütehnilise Ülikooli õppaine PRO1 "Programming I" õpiväljundid. https://www.upc.edu/content/grau/guiadocent/pdf/ing/270001 (03.05.2019)

[46] Kataloonia Polütehnilise Ülikooli õppaine 270012 "Data Structures and Algorithmics" aine kirjeldus. https://www.upc.edu/content/grau/guiadocent/pdf/ing/270012 (03.05.2019)

[47] Konstanzi Ülikooli Informaatika õppekava moodulite raamat. https://www.informatik.uni-konstanz.de/uploads/tx_studiengang/en_modulebook_178_1508485273.pdf (03.05.2019)

[48] Ulmi Ülikooli Informaatika õppekava moodulite raamat https://campusonline.uni-ulm.de/qislsf/pub/modulhandbuecher/wise201819/BA_Informatik_FSPO_2017_WiSe2018_19_Deutsch.pdf?navigationPosition=modulhandbuecher%2CBA-Informatik-FSPO-2017 (03.05.2019)

[49] Bremeni Ülikooli Informaatika õppekava moodulite raamat. https://www.szi.uni-bremen.de/wp-content/uploads/2018/05/mhb_InfB_2018.pdf (03.05.2019)

[50] Johannes Kepleri Ülikool Linzis õppaine [INBIPVOBEKO] "Algorithms and Data Structures" aine kirjeldus. https://studienhandbuch.jku.at/detail.php?lang=de&klaId=INBIPVOBEKO (03.05.2019)

[51] Johannes Kepleri Ülikool Linzis õppaine [INBIPUEALG1] " Berechenbarkeit und Komplexität" aine kirjeldus. http://ssw.jku.at/Teaching/Lectures/Algo/ (03.05.2019)

[52] Genti Ülikooli õppaine C003773 "Algorithms and Data Structures 1" aine kirjeldus. https://studiegids.ugent.be/2019/EN/studiefiches/C003773.pdf (03.05.2019)

[53] Aalto Ülikooli õppaine CS-A1140 "Data Structures and Algorithms" aine kirjeldus. https://mycourses.aalto.fi/course/view.php?id=20548&section=2 (03.05.2019)

[54] Marylandi Ülikooli õppaine CMSC 351 "Algorithms" aine kirjeldus. http://www.cs.umd.edu/~samir/251/251.html (03.05.2019)

[55] Bergeni Ülikooli õppaine INF 102 "Algorithms, Data Structures and Programming" lugemisnimekiri. https://mitt.uib.no/courses/12780/external_tools/118 (03.05.2019)

[56] Princetoni Ülikooli õppaine COS 226 "Algorithms and Data Structures" aine koduleheküIg. https://www.cs.princeton.edu/courses/archive/fall18/cos226/syllabus.php (03.05.2019)

[57] Princetoni Ülikooli õppaine COS 126 "Computer Science: An Interdisciplinary Approach" 6. loeng. http://www.cs.princeton.edu/courses/archive/spring19/cos126/lectures/CS.6.Recursion.2x2.pdf (03.05.2019)

[58] Rensselaeri Polütehnilise Instituudi õppeaine CSCI 1100 "Computer Science I" 23. loeng e-õpikus https://www.cs.rpi.edu/academics/courses/fall18/csci1100/lecture_notes/lec23_recursion.html (03.05.2019)

[59] Bergeni Ülikooli õppaine INF 102 "Algorithms, Data Structures and Programming" 36. nädala 2. loeng. https://mitt.uib.no/courses/12780/files/folder/lecturenotes?preview=1065450 (03.05.2019)

[60] Princetoni Ülikooli õppaine COS 326 "Functional Programming" e-õpik. https://www.cs.princeton.edu/courses/archive/fall18/cos326 (03.05.2019)

[61] Rensselaeri Polütehnilise Instituudi õppeaine CSCI 1200 "Data Structures" 8., 10. ja 13 loeng. https://www.cs.rpi.edu/academics/courses/fall18/csci1200/lectures/08_big_o_notation_recursion.pdf (03.05.2019) https://www.cs.rpi.edu/academics/courses/fall18/csci1200/lectures/10_linked_lists.pdf (03.05.2019) https://www.cs.rpi.edu/academics/courses/fall18/csci1200/lectures/13_problem_solving_I.pdf (03.05.2019)

[62] Aalto Ülikooli õppaine CS-A1120 "Programming 2" e-õpiku peatükk „Recursively defined data structures“. http://a1120.cs.aalto.fi/notes/round-recursion--rdds.html (03.05.2019)

[63] Aalto Ülikooli õppaine CS-A1120 "Programming 2" e-õpiku peatükk „Solving problems recursively“. http://a1120.cs.aalto.fi/notes/round-recursion--solving.html (03.05.2019)

[64] California Ülikooli õppaine CS 61 A "The Structure and Interpretation of Computer Programs" e-raamatu peatükk „Recursive Objects”. http://composingprograms.com/pages/29-recursive-objects.html (03.05.2019)

[65] Aalto Ülikooli õppaine S-E3190 aine "Principles of Algorithmic Techniques" 4. loeng. https://mycourses.aalto.fi/pluginfile.php/516418/mod_resource/content/7/CS-E3190_lect04.pdf (03.05.2019)

[66] Tartu Ülikooli õppeaine "Algoritmid ja andmestruktuurid" 2008 sügise loenguslaidid. https://kodu.ut.ee/~nestra/mat/inf/p/aa/08s/s/graafid.pdf (03.05.2019)

[67] California Ülikooli õppaine CS 61 B "Data Structures" 3. loeng. https://docs.google.com/presentation/d/1c1AQqfCDh2znyIkJM45N2zUq9ww_NN-2Gi9JQkyRROs/edit - slide=id.g825e60afb_0_76 (03.05.2019)

[68] Chalmersi Ülikooli õppaine TDA555 "Introduktion till funktionell programmering" 1. ja 2. loeng. http://www.cse.chalmers.se/edu/year/2018/course/TDA555/Material/Recursion/slides.pdf, http://www.cse.chalmers.se/edu/year/2018/course/TDA555/Material/DataTypes1/slides-2015.pdf (03.05.2019)

[69] California Ülikooli õppaine CS 61 A "The Structure and Interpretation of Computer Programs" kursuse kodulehekülg https://cs61a.org/ (03.05.2019)

[70] Bremeni Ülikooli õppaine BA-700.03 "Praktische Informatik 3: Funktionale Programmierung" 1., 2. ja 8. praktikum. http://www.informatik.uni-bremen.de/~clueth/lehre/pi3.ws18/ueb/uebung-01.pdf, http://www.informatik.uni-bremen.de/~clueth/lehre/pi3.ws18/ueb/uebung-02.pdf, http://www.informatik.uni-bremen.de/~clueth/lehre/pi3.ws18/ueb/uebung-08.pdf, (03.05.2019)

[71] Princetoni Ülikooli õppaine COS 126 "Computer Science: An Interdisciplinary Approach" 6. praktikum. https://docs.google.com/presentation/d/1RIdUvwhMRb1pN5O040wkXA3l2QKlgvtKWtxjkrwsPvc/edit#slide=id.g4397ec55f0_0_1 (03.05.2019)

[72] Helsinki Ülikooli õppaine TKT20001 "Tietorankenteet ja algoritmit" kombineeritud loenguslaidid. https://moodle.helsinki.fi/mod/resource/view.php?id=1494225 (03.05.2019)

[73] E-õpik "Learn You a Haskell for Good!". http://learnyouahaskell.com/recursion#thinking-recursively (03.05.2019)

[74] Marylandi Ülikooli õppaine CMSC 330 "Organization of Programming Languages" 11. loeng. http://www.cs.umd.edu/class/spring2018/cmsc330/lectures/11-tailrecursion.pdf (03.05.2019)

[75] California Ülikooli õppaine CS 61 AS e-õpiku peatükk "Common Recursive Patterns". https://berkeley-cs61as.github.io/textbook/common-recursive-patterns.html (04.05.2019)

[76] Chalmersi Ülikooli õppaine TDA555 "Introduktion till funktionell programmering" 1. praktikum. http://www.cse.chalmers.se/edu/year/2018/course/TDA555/lab1.html (04.05.2019)

[77] Massachusettsi Tehnoloogiainstituudi õppaine 6.0001 " Introdution to Computer Science and Programming in Python" 4. probleemihulk. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/assignments/ (04.05.2019)

[78] Chalmersi Ülikooli õppaine X DIT 961 "Datastrukturer" kursusematerjalid. http://www.cse.chalmers.se/edu/year/2018/course/DIT961/resources/ (04.05.2019)

[79] California Ülikooli õppaine CS 61 A "The Structure and Interpretation of Computer Programs" 3. diskussioon. https://cs61a.org/disc/disc03.pdf (04.05.2019)

[80] California Ülikooli õppaine CS 61 A "The Structure and Interpretation of Computer Programs" 10. loeng. https://docs.google.com/presentation/d/1UnAq6kfCV1QmcN1_xGxQdJl7Nlpe-sx1YbACzLDuGL0/edit#slide=id.p5 (04.05.2019)

[81] Bergeni Ülikooli õppaine INF 101 "Object-oriented programming" 6. praktikum. https://retting.ii.uib.no/inf101.v17.oppgaver/inf101.v17.lab6/blob/master/src/inf101/v17/lab6/recursion/Basic.java (04.05.2019)

[82] Marylandi Ülikooli õppaine CMSC 330 "Organization of Programming Languages" projekt „2A: OCaml Warmup“. https://github.com/anwarmamat/cmsc330spring18-public/tree/master/p2a (04.05.2019)

[83] Princetoni Ülikooli õppaine COS 126 "Computer Science: An Interdisciplinary Approach" e-raamat. https://introcs.cs.princeton.edu/java/23recursion/ (04.05.2019)

[84] Princetoni Ülikooli õppaine COS 126 "Computer Science: An Interdisciplinary Approach" Sierpinski kolmnurga kodutöö. https://introcs.cs.princeton.edu/java/assignments/sierpinski.html (04.05.2019)

[85] Rensselaeri Polütehnilise Instituudi õppeaine CSCI 2300 "Introduction to Algorithms" 3. praktikum. https://www.cs.rpi.edu//~moorthy/Courses/CSCI2300/lab3-2015.html (04.05.2019)

[86] Massachusettsi Tehnoloogiainstituudi õppaine 6.0001 " Introdution to Computer Science and Programming in Python" 6. loeng. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec6.pdf (04.05.2019)

[87] Aalto Ülikooli õppaine A1120 "Programming 2" e-õpiku peatükk „Linked lists“. http://a1120.cs.aalto.fi/notes/round-recursion--lists.html (04.05.2019)

[88] Näidiskood portaalist Geeks for Geeks. https://www.geeksforgeeks.org/count-ways-reach-nth-stair/ (04.05.2019)

[89] Rensselaeri Polütehnilise Instituudi õppeaine CSCI 1100 "Computer Science I" 23. loengu harjutused. https://www.cs.rpi.edu/academics/courses/fall18/csci1100/lecture_notes/lec23_recursion_exercises/exercises.html (04.05.2019)

[90] Princetoni Ülikooli õppaine COS 326 "Functional Programming" 6. probleemihulk. https://www.cs.princeton.edu/courses/archive/fall18/cos326/ass/a6.php (04.05.2019)

[91] Chalmersi Ülikooli õppaine DIT961 "Datastrukturer" 11. loeng. http://www.cse.chalmers.se/edu/year/2018/course/DIT961/files/lectures/dit961_lecture_11.pdf (04.05.2019)

[92] Xkcd veebikoomiksid. https://xkcd.com/710/ (04.05.2019)

[93] Haskelli ülesannete kogum. https://wiki.haskell.org/99_questions/1_to_10 (04.05.2019)

[94] Tartu Ülikooli õppaine "Programmeerimise alused II" kursuse koduleht. https://courses.cs.ut.ee/2017/eprogalused2/spring/Main/PART6Sr (04.05.2019)

[95] Carnegie Mellon Ülikooli kursuse "Programming in Standard ML" sügis 2000 veebilehekülg. https://www.cs.cmu.edu/~rwh/introsml/techniques/memoization.htm (04.05.2019)

[96] Indiana South Bend Ülikooli kursuse C311 "Programming Languages" 2019 veebilehekülg. http://www.cs.iusb.edu/~dvrajito/teach/c311/c311_6_deeprec.html (04.05.2019)

[97] Tartu Ülikooli kursuse "Algoritmid ja andmestruktuurid" 1. arvutipraktikum. http://kodu.ut.ee/~kiho/ads/Praktikum/Arvutipraktikum/2-KMB/Sissejuhatus/Peafail.pdf (04.05.2019)

[98] A. Pederi, J. Kiho ja H. Nestra „Algoritmid ja andmestruktuurid. Ülesannete kogu“ (ISBN: 978-9949-77-377-0)

Lisad

I. Rekursiooni käsitlevad õppeained Tartu Ülikooli õppekavades

Bakalaureusetaseme õpe

Tabel 1: Rekursiooni käsitlevad õppeained Tartu Ülikooli bakalaureusetaseme õppekavades

	Sihtrühm	**Informaatika 2476**	**Arvutitehnika 83866**	**Matemaatika 2472**	**Matemaatiline statistika 2474**	**Teised õppekavad**
Programmeerimine 6 EAP LTAT.03.001	informaatika bakalaureuse 1. aasta üliõpilased (kui on läbimata aine MTAT.03.236)	+	+	+	+ (või MTAT.03.256)	Füüsika, keemia ja materjaliteadus (144301)
Programmeerimise alused II 3 EAP MTAT.03.256	*This course could be taken as elective for students of curricula outside of the Institute of Computer Science*	-	-	-	+ (või LTAT.03.001)	Eesti ja soome-ugri keeleteadus (2416) (valitav erialamoodul) Loodus- ja reaalainete õpetamine põhikoolis (144897) (valitav suunamoodul) Majandusteadus (2442) (valikaine)
Objekt-orienteeritud programmeerimine 6 EAP LTAT.03.003	-	+	+ (valitav suunamoodulis)	+	+ (valitav suunamoodulis)	-
Algoritmid ja andmestruktuurid 6 EAP LTAT.03.005	informaatika bakalaureuse 2. aasta üliõpilased	+	+ (valitav suunamoodulis)	+ (valitav suunamoodulis)*	+ (valitav suunamoodulis)	-
Automaadid, keeled ja translaatorid 6 EAP LTAT.03.006	informaatika bakalaureuse 2. aasta üliõpilased	+ (valitav suunamoodulis)	-	+ (valitav suunamoodulis)*	-	-
Programmeerimiskeeled 6 EAP MTAT.03.006	informaatika bakalaureuse 3. aasta üliõpilased	+ (valikaine)	+ (valikaine)	-	-	-

* soovituslik üliõpilastele, kes kavatsevad õpinguid jätkata informaatika magistriõppes (eriti teoreetilise informaatika, krüptograafia, andmekaeve suunal).

Rakenduskõrgharidusõpe

Tabel 2: Rekursiooni käsitlevad õppeained Tartu Ülikooli rakenduskõrgharidusõppe õppekavas

	Infotehnoloogiliste süsteemide arendus (136638)
Programmeerimine 6 EAP LTAT.03.001	+
Objektorienteeritud programmeerimine 6 EAP LTAT.03.003	+

Magistri- ja doktorikraadi õpe

Tabel 3: Rekursiooni käsitlevad õppeained Tartu Ülikooli magistri- ja doktoriõppe õppekavades

	Sihtrühm	**Infotehnoloogia mitteinformaatikutele (144919)**	**Küberkaitse (100946)**	**Arvutitehnika ja robootika (136637)**	**Informaatika (129537)**	**Varia**
Programmeerimine 6 EAP LTAT.03.001	Küberkaitse magistriõppe tudengitele	+	+			Geenitehnoloogia (2597) (kohustuslik, kui valitakse Bioinformaatika suund)
Programmeerimise alused II 3 EAP MTAT.03.256	bakalaureus, aga valitav magistris					Eesti ja soome-ugri keeleteadus (2543) (valikaine) Keskkonnatehnoloogia (2590) (valikmooduli osa) Matemaatika- ja informaatikaõpetaja (2501) * Psühholoogia (2574) (kohustuslikus moodulis valik) Põhikooli mitme aine õpetaja (2498) (valitava mooduli osa) Religiooniuuringud ja teoloogia (194378) (valikaine)
Objektorienteeritud programmeerimine 6 EAP LTAT.03.003	Aine on IT mitteinformaatikutele magistriõppe üliõpilastele	+ (valikaine)				
Algoritmide kavandamine ja analüüs 6 EAP MTAT.03.286	magistriõpe, doktoriõpe		+ (valikkursus)	+ (erialamooduli valikmoodul)	+	Informaatika (80333) (Dokt) (valikaine)
Algoritmika 6 EAP MTAT.03.238	magistriõpe			(erialamooduli valikmoodul)	+	Tarkvaratehnika (100864) (valikaine)

* kohustuslik matemaatika- ja informaatikaõpetaja erialal

II. Väljapakutud ülesannete komplekt

Rekursiooni mõiste tundmine ja lihtsama rekursiooni realiseerimine

Soovitus: proovi ülesannete juures esmalt „määrata tingimusi, kui rekursiooni ei pea kasutama" [32].

Loe esmalt üle küsimused, mis aitavad Sind tulevaste põnevate proovikivide lahendamisel. Kui satud hätta, siis mõtle neid uuesti rahulikult läbi [61:13]:

Ideede loomine ja hindamine [61:13]:

1. Kõige olulisem on näidetega katsetada! Kas Sa suudaksid mõelda välja strateegia selle probleemi lahendamiseks? Katseta igat strateegiat mitmel näitel. Ehk on võimalik seda strateegiat enne koodi kõigepealt algoritmina kirjutada?
2. Proovi esmalt lahendada lihtsamat versiooni probleemist ning siis kas õpi harjutusest või üldista tulemust.
3. Kas see probleem meenutab mõnda probleemi, mida Sa juba oskad lahendada?
4. Kui keegi annaks Sulle osalise lahenduse, siis kas sa oskaksid seda laiendada täislahenduseks?
5. Mis juhtuks, kui sa poolitaksid probleemi ja üritaksid mõlemat poolt rekursiivselt eraldi lahendada?
6. Kas andmete sorteerimine aitaks?
7. Kas Sa saad probleemi jagada erijuhtudeks ning igat juhtu eraldi käsitleda?
8. Kas Sa saaksid leida midagi fundamentaalset selle probleemi juures, mis aitaks seda lihtsamalt või efektiivsemalt lahendada?
9. Kui Sul on idee olemas, mis Sa arvad, et töötab, siis peaksid hindama: kas see tõesti töötab? kas leidub paremaid/ kiiremaid lähenemisi? kui see ikkagi ei tööta, siis miks?

Ideede kajastamine koodis [61:13]:

1. Kuidas Sa kavatsed andmeid esindada? Milline viis on lihtsaim ja milline efektiivseim?
2. Kas Sa saad kasutada klasse andmete organiseerimiseks? Milliseid andmeid peaksid koos hoiustama ja manipuleerima? Millist informatsiooni on igal objektil tarvis? Millised operatsioonid (peale lihtsamate get() ja set() meetodite) võiksid olla abiks?

3. Kuidas saaksid probleemi loogilisteks üksusteks jagada, millest hiljem kasvaksid välja funktsioonid? Kas Sa saad taaskasutada koodi, mida oled juba kirjutanud? Kas mõni loogika, mida Sa praegu kirjutad, on taaskasutatav ka hiljem?
4. Kas Sa kasutad rekursiooni või iteratsiooni? Millist informatsiooni pead läbi tsüklite või rekursiivsete funktsioonikutsete edastama ja kuidas seda "edastatakse"?
5. Kui efektiivne on Sinu lahendus? Kas see on üldine? Kas suudad nüüd mõelda paremaid ideid või lähenemisi?
6. Tee endale koodi loogikat kirjutades märkmeid. Nendest märkmetest saavad muutumatud väärtused (ingl *invariants*) st. need mõtted, mis peaksid olema tõesed iga iteratsiooni/ rekursiivse funktsioonikutse korral.

Detailid [61:13]:

1. Kas kõik on korrektselt algväärtustatud?
2. Kas Sinu tingimuslausete loogika on korrektne? Kontrolli paar korda üle ja katseta näidetega ka käsitsi.
3. Kas Sinu tsüklite piirid on korrektsed st. kas peaksid lõpetama kui väärtus on n, n-1 või n-2?
4. Puhasta oma koodi ja loogika kommentaarid.
5. Kas Su kood toimib muu hulgas äärejuhtudega (nt. kui vastus on andmete alguses, andmetes leidub korduvaid väärtusi või andmestik on väga väike/ suur)?

Ülesanded

1. Arvude 1 kuni n summa.

Tehnilised oskused: (8) baasjuhu ja sammu äratundmine ning mõistmine, (10) rekursiooni vähenemine baasjuhu suunas, (15) algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek.

Teosta funktsioon *summa*, mis summeerib rekursiivselt kõik arvud 1-st sisendparameeter *n*-ini [70:1]. Funktsiooni koostamisel võib Sulle abiks olla joonisel 1 nähtav summa matemaatiline definitsioon:

$$summa(n) = \sum_{i=1}^{n} i$$

Joonis 1: Summa matemaatiline definitsioon [70:1].

2. Naturaalarvu ristsumma.

Tehnilised oskused: (8) baasjuhu ja sammu äratundmine ning mõistmine, (10) rekursiooni vähenemine baasjuhu suunas, (11) funktsiooni käitumine ootamatute argumentide korral, (15) algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek.

Ristsumma on summa [70:2, 81], mis saadakse selle arvu kümnendesituses esinevate numbrimärkide kokku liitmisel (nt. 457=4+5+7=16). Me saame ristsummat arvutada rekursiivselt, kui ristsumma(x) on numbri x ristsumma ja x/y on arvu x täisarvuline jagamine y-iga:

$$ristsumma(x) = \begin{cases} 0, & kui\ x = 0; \\ x\ mod\ 10 + ristsumma(x \div 10), & vastasel\ juhul. \end{cases}$$

kus *mod* on jääk, mis tekib arvu x jagamisel arvuga 10.

Joonis 2: Ristsumma matemaatiline definitsioon [81].

Teosta rekursiivne funktsioon ristsumma. Testige võimalusel pinginaabriga üksteise lahendusi (ka erijuhtudega). Pinginaabriga lahendust vahetades saad võimaluse tutvuda teise perspektiiviga ning seeläbi enda lahendust võib-olla täiustada.

Vihje: funktsioon peab töötama negatiivsete arvude korral, aga ei pea töötama ujukomaarvude korral.

3. Tarvilike ehitusplokkide arv olenevalt püramiidi kõrgusest [81]

Tehnilised oskused: (8) baasjuhu ja sammu äratundmine ning mõistmine, (9) alam-ülesandest terve ülesande lahenduseni jõudmine, (11) funktsiooni käitumine ootamatute argumentide korral.

Kujuta ette püramiidi, mis on ehitatud võrdsete suurustega ruudukujulistest ehitusplokkidest. Püramiidi iga korrus koosneb plokkide ruudustikust. Kõige ülemisel korrusel on üks plokk, teisel korrusel 2x2 plokki, kolmandal 3x3 jne.

Meie ülesandeks on olenevalt püramiidi kõrgusest tagastada selle plokkide koguarv. Lahenda ülesanne rekursiivselt, kasutamata ühtegi tsüklit. Kui püramiidi kõrgus on negatiivne, siis selle asemel, et püramiidi maapinnast kõrgemaks ehitada, tahavad insenerid hoopis kaevata püramiidikujulise augu. See tähendab, et plokke jääb üle ja vastus peaks olema negatiivne.

4. Järjendi elementide summa kuni parameetrina antud indeksini [82]

Tehnilised oskused: (8) baasjuhu ja sammu äratundmine ning mõistmine, (10) rekursiooni vähenemine baasjuhu suunas, (11) funktsiooni käitumine ootamatute argumentide korral, (16) fikseeritud tasemete arvuga andmestruktuuri töötlemine.

Kirjuta funktsioon osa_summa(x, arr), mis võtab parameetritena indeksi *x* ja järjendi *arr* ning tagastab järjendi elementide summa kuni indeksini *x*. Eeldame, et tegu on ühetasemelise järjendiga. Testi oma lahendust muu hulgas erijuhtudega.

Näide: osa_summa(2, [5; 6; 7; 3]) = 5+6+7 = 18.

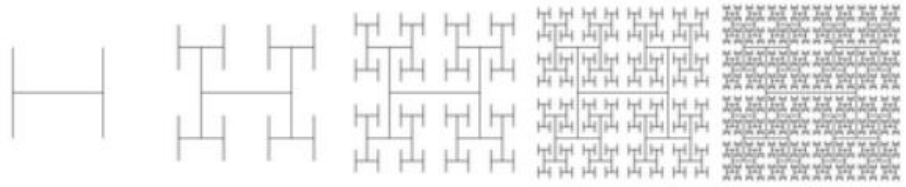
Fraktaalid

Fraktaal on Princetoni Ülikooli definitsiooni kohaselt [83] geomeetriline kujund, mida saab lõpmatult osadeks jagada. Iga selline osa on vähendatud suurusega koopia originaalsest kujundist. Fraktaalid on loomult seega iseendale viitavad struktuurid, mis aitavad omakorda vältida koodikordust.

5. H-puu [83]

Tehnilised oskused: (3) puurekursioon (sh. binaarne rekursioon), (8) baasjuhu ja sammu äratundmine ning mõistmine, (9) alamülesandest terve ülesande lahenduseni jõudmine, (10) rekursiooni vähenemine baasjuhu suunas, (14) koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine.

Kirjuta funktsioon n-tasemelise H-puu joonistamiseks. Vaata joonis 3-e ning püüa mõelda juhu peale, kui rekursiooni ei pea enam kasutama (st. baasjuht) ning sellele, milline osa jooniselt kordub st. mida peaks kasutama rekursiivse(te) funktsioonikutse(te)na.



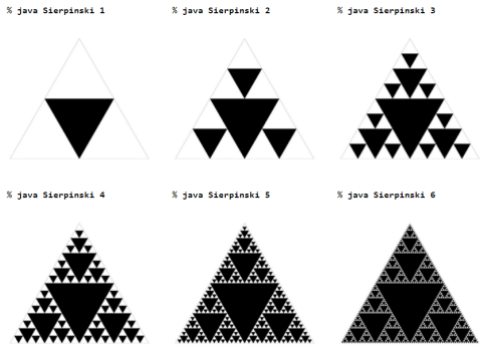
Joonis 3: n-tasemeline H-puu [83].

Kui Sinu funktsioon näeb sisendi n=5 korral samasugune välja, kui joonisel 3 nähtav parempoolseim joonistus, siis proovi oma funktsioonis rekursiivse(te) funktsioonikutse(te) ja baasjuhu asukohad vahetada. Mõtesta endale lahti, mis juhtus.

6. Sierpinski kolmnurk

Tehnilised oskused: (3) puurekursioon (sh. binaarne rekursioon), (8) baasjuhu ja sammu äratundmine ning mõistmine, (9) alamülesandest terve ülesande lahenduseni jõudmine, (15) algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek.

Teosta funktsioon n-tasemelise Sierpinski kolmnurga joonistamiseks. Sierpinski mustrit kirjeldas Princetoni Ülikooli sõnul [84] aastal 1915 Poola matemaatik Waclaw Sierpinski, kuid seda on esinenud Itaalia kunstis juba 13. sajandil. Olgugi, et kolmnurk näeb keeruline välja, siis saab teda genereerida lihtsa rekursiivse funktsiooniga. Mõtle rekursiivselt: Sinu funktsioon peaks joonistama ühe musta kolmnurga (suunaga alla) ning siis koos sobiva peatumisjuhuga kutsuma iseennast 3 korda välja. Programmi kirjutades harjuta modulaarset disaini st. loo mitterekursiivne funktsioon kolmnurk(), mis joonistab sisendparameetri n suuruse võrdkülgse kolmnurga [84]. Joonis 4 illustreerib, milline Sinu joonistatud kolmnurk peaks olema olenevalt sisendtasemest (joonisel kuni tasemeni n=6):



Joonis 4: Sierpinski kolmnurk olenevalt tasemest *n* [84].

Ingliskeelne lisalugemine:
https://introcs.cs.princeton.edu/java/assignments/sierpinski.html

Kognitiivsete oskuste arendamine

7. Grupiarutlus

Tehnilised oskused: (8) baasjuhu ja sammu äratundmine ning mõistmine, (10) rekursiooni vähenemine baasjuhu suunas, (12) rekursiivse funktsiooni olemus (nt. rekursiivne funktsioon ei pea alati midagi tagastama ega edastama).

Leia partner või grupp; arutage läbi kolm järgnevat küsimust [71]:

1. Mida see tähendab kui öeldakse, et programm kasutab rekursiooni?
2. Mis on rekursiooni kaks komponenti?
3. Kuidas erineb Collatzi algoritm enamikest teistest rekursiivsetest algoritmides? Otsige kõigepealt informatsiooni algoritmi kohta. Kui jääte hätta, siis lugege lisa siit: https://introcs.cs.princeton.edu/java/23recursion/.

8. Funktsiooni mõttes väärtustamine I [83]

Tehnilised oskused: (7) mõttes etteantud funktsiooni järgi avaldise väärtustamine olenevalt sisendist, (8) baasjuhu ja sammu äratundmine ning mõistmine.

Vaatle järgmist rekursiivset funktsiooni [83]:

```
public static int mystery(int a, int b) {
    if (b == 0)       return 0;
    if (b % 2 == 0)   return mystery(a+a, b/2);
    return mystery(a+a, b/2) + a;
}
```

a. Väärtusta peas. Mis on mystery(2,25) väljund?
b. Väärtusta peas. Mis on mystery(3,11) väljund?
c. Olgu a ja b täisarvud. Selgita, mis väärtust mystery(a,b) arvutab.

Vastused: 1: 50, 2: 33, 3: a*b.

9. Funktsiooni mõttes väärtustamine II [11]

Tehnilised oskused: (7) mõttes etteantud funktsiooni järgi avaldise väärtustamine olenevalt sisendist, (8) baasjuhu ja sammu äratundmine ning mõistmine, (10) rekursiooni vähenemine baasjuhu suunas, (13) rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast, (18) rekursiooni mälukasutus.

Väärtusta peas järgmine funktsioon sisendparameetriga 4 [11]:

```
def prindiRuudud(ylemLimiit: Int): Unit = {
 if (ylemLimiit > 0) {
    prindiRuudud(ylemLimiit)
    println(ylemLimiit*ylemLimiit)
 }
}
```

Millised järgnevatest väidetest [11] kirjeldavad täpselt olukorda, mis juhtub? Vali kõik, mis on tõesed:

- □ Kood ei prindi välja ühegi arvu ruutu.
- □ Kompilaator kaebab. Programmi ei saa jooksutada.
- □ Programm jookseb „ühe koha peal“ st. olgugi, et ta ei nõua palju mälu, siis ta ei lõpe.
- □ Programm tarbib senikaua lisamälu kuni seda enam ei ole st. programm üritab enne kokku jooksmist luua funktsioonikutsete magasini „lõpmatu“ arv kaadreid.
- □ Koodis ei ole defineeritud baasjuhtumit. Seetõttu jõuab funktsioon negatiivsete väärtustega parameetriteni.
- □ Eksisteerib baasjuht, aga programm ei jõua kunagi selleni.

10. Funktsiooni mõttes väärtustamine III [11]

Tehnilised oskused: (7) mõttes etteantud funktsiooni järgi avaldise väärtustamine olenevalt sisendist, (8) baasjuhu ja sammu äratundmine ning mõistmine, (10) rekursiooni vähenemine baasjuhu suunas, (13) rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast, (18) rekursiooni mälukasutus.

Väärtusta peas järgmine funktsioon sisendparameetriga 4 [11]:

```
def prindiRuudud(ylemLimiit: Int): Unit = {
       prindiRuudud(ylemLimiit – 1 )
       println(ylemLimiit*ylemLimiit)
}
```

Millised järgnevatest väidetest [11] kirjeldavad täpselt olukorda, mis juhtub? Vali kõik, mis on tõesed:

- □ Kood ei prindi välja ühegi arvu ruutu.
- □ Kompilaator kaebab. Programmi ei saa jooksutada.
- □ Programm jookseb „ühe koha peal“ st. olgugi, et ta ei nõua palju mälu, siis ta ei lõpe.
- □ Programm tarbib senikaua lisamälu kuni seda enam ei ole st. programm üritab enne kokku jooksmist luua funktsioonikutsete magasini „lõpmatu“ arv kaadreid.
- □ Koodis ei ole defineeritud baasjuhtumit. Seetõttu jõuab funktsioon negatiivsete väärtustega parameetriteni.

□ Eksisteerib baasjuht, aga programm ei jõua kunagi selleni.

11. Funktsiooni mõttes väärtustamine IV [85]

Tehnilised oskused: (5) mitmekordne rekursioon, (7) mõttes etteantud funktsiooni järgi avaldise väärtustamine olenevalt sisendist, (8) baasjuhu ja sammu äratundmine ning mõistmine, (10) rekursiooni vähenemine baasjuhu suunas, (13) rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast.

Kui mitu rida prindib järgmine programm [85]? Võid eeldada, et sisend *n* on kahe aste. Ürita leida ridade arv 1, 2, 4, 8 korral. Kas näed mustrit?

```
function f(n):
	if (n>1):
		print_line("veel töötan")
		f(n/2)
		f(n/2)
```

Vastus: *n*=1 korral 0 rida, *n*=2 korral 1 rida, *n*=4 korral 3 rida ja *n*=8 korral 7 rida.

Rekursiivse funktsiooni analüüsimine ja töö käigu selgitamine

12. Põimemeetod

Tehnilised oskused: (3) puurekursioon (sh. binaarne rekursioon), (7) mõttes etteantud funktsiooni järgi avaldise väärtustamine olenevalt sisendist, (8) baasjuhu ja sammu äratundmine ning mõistmine, (10) rekursiooni vähenemine baasjuhu suunas, (15) algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek, (22) aktiveerimiskirjete ja/või rekursioonipuu koostamine.

Põimemeetod on fundamentaalselt rekursiivne ning selle rekursiivse sammu põhiliseks eesmärk on põimimisprotsess [58]. Samas toob Bergeni ülikool oma loengus välja, et põimemeetodi probleeme saab lahendada ka iteratiivselt [59].

Põimimine viitab kahe sorteeritud (väiksema) listi põimimisele suurde sorteeritud listi. Kõnealune „suur list“ koostatakse kandes üle väikeste listide väikseimad numbrid senikaua suurde listi, kuni väikesed listid on tühjad.

Põimemeetodi tööst saab mõelda järgnevalt [59]:

1. Kui järjendis on 1 või 0 elementi, siis oleme juba valmis.
2. Poolita järjend keskelt. Kutsugem vasakut järjendit A-ks ja paremat B-ks.
3. Korda eelnevaid samme mõlema järjendiga kuni täidame baasjuhtu:
   - üks kord järjendiga A;
   - üks kord järjendiga B.
4. Põimi listid A ja B üheks „suureks listiks“ kandes üle väikeste listide väikseimad numbrid senikaua suurde listi, kuni väikesed listid on tühjad.

Küsimus [59]: millal rekursiooni peatada?

Joonista rekursioonipuu järjendi [9, 1, 4, 1, 5, 8] sorteerimiseks põimemeetodiga. Selgita meetodi tööd pinginaabrile või sõbrale. Kui põimemeetod tundub veel segane, siis vaata julgelt videot [59]: https://www.youtube.com/watch?v=4VqmGXwpLqc.

13. Palindroomide rekursioonipuu ning pseudokood

Tehnilised oskused: (3) puurekursioon (sh. binaarne rekursioon), (7) mõttes etteantud funktsiooni järgi avaldise väärtustamine olenevalt sisendist, (8) baasjuhu ja sammu äratundmine ning mõistmine, (10) rekursiooni vähenemine baasjuhu suunas, (11) funktsiooni käitumine ootamatute argumentide korral, (15) algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek, (22) aktiveerimiskirjete ja/või rekursioonipuu koostamine.

Vaatleme rekursiooni tööd väljaspool arvandmeid. Palindroom on „sõna (v. lause), mis tagurpidi loetuna annab sama sõna (v. lause)“[4]. Palindroomi äratundmise algoritm on MIT sõnul järgnev [86]:

1. Eemalda sõnest kõik peale väiketähtede st. kirjavahemärgid ja suurtähed.
2. Sõne, mille pikkus on 0 või 1 on palindroom.
3. Kui sõne esimene täht võrdub sõne viimase tähega ja sama kehtib ka iga kord paarina esimese ja viimase tähe eemaldamisel (ning kontrollimisel), siis on sõne palindroom.

[4] [EKSS] "Eesti keele seletav sõnaraamat" https://www.eki.ee/dict/ekss/index.cgi?Q=palindroom&F=M.

Otsi internetist mõni vahva palindroom ja joonista selle kontrollimise kohta rekursioonipuu. Kirjuta selle põhjal omakorda rekursiivne pseudokood funktsioonist palindroom(), mis võtab parameetrina sisse sõne. Mõtle muu hulgas juhule, kuidas peaks funktsioon toimima, kui parameetriks ei ole palindroom.

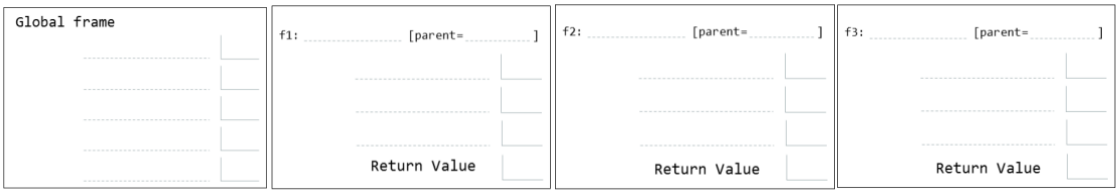
14. Aktiveerimiskirjed

Tehnilised oskused: (8) baasjuhu ja sammu äratundmine ning mõistmine, (13) rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast, (22) aktiveerimiskirjete ja/või rekursioonipuu koostamine, (23) rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine.

Joonista järgmise meetodi jaoks aktiveerimiskirjed [79] :

```
def rec(x, y):
    if y > 0:
        return x * rec(x, y - 1)
    return 1
rec(3, 2)
```

Kasuta järgmist joonisel 5 toodud malli:



Joonis 5: Mall funktsiooni läbimängu jaoks programmeerimiskeskkonna seisukohast [79].

Boonusküsimus: mida see funktsioon teeb?

NB! California Ülikool toob välja [79], et selle probleemi lahendamine rekursiivselt ei ole tegelikult hea näide rekursioonist, sest iteratiivne teostus on kiirem ja kasutab vähem mälu. See probleem on mõeldud aitamaks Sul aru saada, mis programmeerimiskeskkonnas juhtub, kui me usaldame rekursiooni pimesi (ingl *leap of faith*). Proovi silumisel (ingl *debugging*) vältida taolist käsitsi läbimängu.

15. Eukleidese algoritm

Tehnilised oskused: (8) baasjuhu ja sammu äratundmine ning mõistmine, (11) funktsiooni käitumine ootamatute argumentide korral, (15) algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek.

„Eukleidese algoritm leiab kahe naturaalarvu suurima ühisteguri (gcd ehk ingl *greatest common divisor*). Algoritm on järgmine [13]:

- Olgu meil naturaalarvud a ja b ning on teada, et a > b.
- Kui b = 0, siis on suurim ühistegur a.
- Kui ei, siis korda protsessi, võttes uueks a-ks b ja uueks b-ks endiste a ja b jagamisel saadud jääk.

Näide: „gcd(102, 68) = 34“ [83].

Realiseeri Eukleidese algoritm rekursiivse funktsioonina nii, et programm töötab kasutaja sisendiga. Testi programmi tööd erinevate sisendite korral.

16. Elemendi olemasolu lihtahelas

Tehnilised oskused: (4) lineaarne- ja sabarekursioon, (11) funktsiooni käitumine ootamatute argumentide korral, (15) algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek, (23) rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine.

Mängi läbi ja uuri esmalt lihtahelat interaktiivselt: https://visualgo.net/en/list.
Teosta sabarekursiivne meetod kontrollimaks, kas lihtahel sisaldab elementi *e*. Lihtahel on definitsioonilt rekursiivne andmestruktuur [61:10, 62, 64] kuna „lihtahel on kas tühi või sisaldab tippu, milles on väärtus ja viit teisele lihtahelale” [61:10].

Kirjuta funktsioon, mis võtab sisendiks lihtahela ning elemendi ning kontrollib selle elemendi olemasolu lihtahelas. Probleemist saab mõelda järgnevalt [87]:

1. Tühi järjend ei sisalda ühtegi elementi.
2. Kui järjendi esimene element on *e*, siis järjend sisaldab seda elementi.
3. Vastasel juhul sisaldab järjend elementi siis, kui järjendi saba (ingl *tail*) seda sisaldab.

Testi enda funktsiooni erijuhtudega ja vajadusel loe lisa Aalto Ülikooli materjalist, mis käsitleb lihtahelat FP raames: http://a1120.cs.aalto.fi/notes/round-recursion--lists.html.

Rekursiivse funktsiooni muutmine, täiendamine ja edasi arendamine

17. Erinevaid viise minna üles trepist [79]

Tehnilised oskused: (5) mitmekordne rekursioon, (8) baasjuhu ja sammu äratundmine ning mõistmine (14) koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine.

Sa tahad minna üles trepist, millel on n astet. Korraga saad teha üks või kaks sammu. Kui mitmel eriviisil saad sellisest trepist üles minna? Eeldame, et n on naturaalarv [79].

```
def abifun(n):
      if n <= 1:
            return n
      return abifun(n-1) + abifun(n-2)

def viise(n):
      return abifun(n+1)

print(viise(3))
```

Joonis 6: näidislahendus [88].

Uuri joonisel 6 nähtavat koodi, mis rakendab mitmekordset rekursiooni. Mille arvutamise moodi näeb välja funktsioon abifun()? Kui oled kindel, et said sellest aru, siis muuda funktsiooni nii, et 1 ja 2 sammu asemel võib korraga astuda kuni (k.a.) *k* sammu korraga [79]. Nt. viise(4,4) = 8 ja viise(10,3) = 274.

18. Meeldetuletus [83]

Tehniline oskus: (14) koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine.

Vaata uuesti 8. ülesannet. Asenda etteantud funktsioonis plussmärk korrutamismärgiga ja ’return 0’ ’return 1’ga [83]. Olgu a ja b täisarvud. Selgita, mis väärtust mystery(a,b) arvutab.

Vastus: a^b.

19. Fibonacci

Tehnilised oskused: (6) mitmepoolne rekursioon, (8) baasjuhu ja sammu äratundmine ning mõistmine, (14) koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine, (19) rekursiooni ajaline keerukus.

Fibonacci arvude funktsioon on üks populaarsemaid ülesandeid, mida rekursiooni õppijad lahendavad. Fibonacci arvujärjendi definitsioon on järgnev [58]:

1. Algtingimused: $f(0) = 0$ ja $f(1) = 1$.
2. Rekurretne seos: $f(n) = f(n-1) + f(n-2)$.

Kirjuta meetod, mis arvutab sisendi *n* põhjal *n*-inda Fibonacci arvu.

Vaatame nüüd uuesti Fibonacci arvude definitsiooni. Näeme, et f(n-1) arvutamine nõuab f(n-2) arvutamist, mida nõutakse ka f(n) arvutamisel. Toimub üleliigne arvutamine, mis arvude f(n-3), f(n-4) jne. korral veelgi drastilisem [89]. Kui Sa ei ole selles veendunud, siis joonista funktsioonikutsete puu, mis illustreerib ebaefektiivust.

Lahendus: kui jäid puu joonistamisega hätta, siis uuri Princetoni Ülikooli loengumaterjali 39. ja 40. slaidi: http://www.cs.princeton.edu/courses/archive/spring19/cos126/lectures/CS.6.Recursion.2x2.pdf.

Meie efektiivsusmure parandab dünaamiline programmeerimine ehk mäluga Fibonacci. Dünaamilise programmeerimise põhimõtteks on Bergeni Ülikooli sõnul [65:15] rekursiivselt komplekssе probleemi mitmeks alamprobleemiks jagamine ning iga alamprobleemi vastuse mälus hoidmine.

Muuda enda algset funktsiooni nii, et see kasutaks mälu.

Kui Sa ei ole veel efektiivsuse paranemises veendunud, siis vaata läbi mõlema Fibonacci läbimäng: https://www.cs.usfca.edu/~galles/visualization/DPFib.html

Vihje [90]: Kui Sa ei tea, kust alustada mäluga_fib() kirjutamist, siis üheks heaks strateegiaks on kasutada mitmepoolset rekursiooni (ingl *mutual recursion*). Tee üks funktsioonidest signatuuri poolt nõutavaks Fibonacci funktsiooniks. Teine funktsioon vastutagu aga arvutatud väärtuste salvestamise ja uuendamise eest. Selle strateegia eelis on see, et ta laseb Sul eristada memoiseerimise funktsioonist, mida memoiseeritakse. Memoiseerimine (ingl *memoization)* on programmeerimistehnika, mille raames salvestatakse tehtud arvutused vahemällu, et neid saaks vajadusel hiljem kiiresti ja uuesti

arvutamata välja otsida [95]. Seega on vihjena toodud strateegias Fibonacci arvu arvutav funktsioon ja arvutusi salvestav funktsioon eraldi.

Rekursiivsed andmetüübid ja -struktuurid:

20. Kiirsorteerimine

Tehnilised oskused: (3) puurekursioon (sh. binaarne rekursioon), (23) rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine.

Uurisime ennist põimemeetodit, kui joonistasime 12. ülesandes rekursioonipuu. Bergeni Ülikool toob oma loengus [59] välja, et erinevalt põimemeetodist ei saa kiirsorteerimist iteratiivselt lahendada. Põimemeetod sorteerib suunaga üles, kiirmeetod aga suunaga alla. Põimemeetodit ei saa rakendada järjendisiseselt, kuid kiirmeetodit saab.

Vaata järgmist läbimängu ning õppevideot ja kirjuta kiirsorteerimise algoritm:

1. interaktiivne läbimäng: https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html;
2. õppevideo [59]: https://www.youtube.com/watch?v=ZHVk2blR45Q.

Lahendus: näidislahenduse leiad nt. Chalmersi Ülikooli loengumaterjalidest: http://www.cse.chalmers.se/edu/year/2018/course/DIT961/files/lectures/dit961_lecture_11.pdf.

21. Kaustapuu

Tehnilised oskused: (1) iteratiivselt lahendatud probleemi rekursiooniga lahendamine, (3) puurekursioon (sh. binaarne rekursioon), (8) baasjuhu ja sammu äratundmine ning mõistmine, (17) fikseerimata tasemete arvuga andmestruktuuride töötlemine, (23) rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine.

Kaustapuu suurus. Olgu meil kaustapuu, millel on alamkaustad, millel on omakorda alamkaustad. Teosta rekursiooni kasutamata funktsioon, mis leiab ülemkausta suuruse.

Ilmselt leiad, et kuna failisüsteemid on loomult rekursiivsed andmetüübid (st. iseendale viitavad), siis on rekursioonita lahendus iteratiivsest keerukam [57]. Kirjuta nüüd sama funktsioon rekursiivselt. Arutle kumba lahendust Sa eelistad ja miks.

22. Sabarekursiooni äratundmine

Tehnilised oskused: (4) lineaarne- ja sabarekursioon, (8) baasjuhu ja sammu äratundmine ning mõistmine.

TalTechi õpik õpetab, et sabarekursioon on ihaldusväärne lahendusmeetod. Miks? Kuna meetod lõpeb samaaegselt, kui rekursiivne funktsioonikutse, siis ei ole vajadust hoiustada vahepealseid funktsioonikutseid magasinis. Samuti saavad kompilaatorid sabarekursiooni optimeerida, et magasini suurust (ja sealhulgas mälukasutust) vähendada [10].

Vasta järgmiste funktsioonide põhjal, kas tegu on sabarekursiooniga või mitte [91]:

```
void hello(int n) {
      if (n>0) {
            System.out.println("hello world");
            hello(n-1);
      }
}
```

Vastus [91]: Jah! – midagi muud ei juhtu peale funktsiooni hello() rekursiivset funktsioonikutset.

```
int fac(int n) {
      if (n==0) return 1;
      else return n * fac(n-1);
}
```

Vastus [91]: Ei! – peale fac(n-1) rekursiivset funktsioonikutset peab selle tulemuse veel n-iga korrutama.

23. Meeldetuletus: lihtahel

Tehnilised oskused: (4) lineaarne- ja sabarekursioon.

16. ülesande raames kirjutasime funktsiooni, mis võtab sisendiks lihtahela ning ühe elemendi ja kontrollib selle elemendi olemasolu lihtahelas. Proovi enda lahendusloogikat ilma piilumata meelde tuletada. Kirjuta lahenduse pseudokood.

Lihtahel on üks andmestruktuuridest, mis on loomult rekursiivne. Lihtahelat rakendavate probleemide lahendused on üldiselt rekursiivsed. Kui Sulle Sinu eelnev lahendusloogika hästi meelde ei tule, siis proovi 16. ülesannet uuesti lahendada. Kordame seda, et jätta meelde andmestruktuure, nagu lihtahel ja puu, mille rakendus probleemilahendustes nõuab rekursiooni kasutust.

Rekursiivse funktsiooni efektiivsuse võrdlemine teiste lahendusmeetoditega ja eelistuse tegemine

24. Fibonacci – mäluga vs. iteratiivne

Tehnilised oskused: (18) rekursiooni mälukasutus ja ajaline keerukus, (20) iteratiivse ja rekursiivse koodi ajalise keerukuse vahe, (21) iteratiivse ja rekursiivse koodi mälukasutuse vahe, (23) rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine.

Teostasime ülesandes 19. mäluga Fibonacci funktsiooni. Kirjuta nüüd iteratiivne Fibonacci funktsioon. Trikk on ehitada lahendus ülesse kasutades for-tsüklit, mis arvutab esimesena f2, siis f3, siis f4 jne [89].

Võrdle mäluga ja iteratiivse Fibonacci efektiivsust ja ajakulu kasvavate sisendite korral. Kumba lahendust eelistad ja miks?

25. Mäluga Collatzi hüpotees

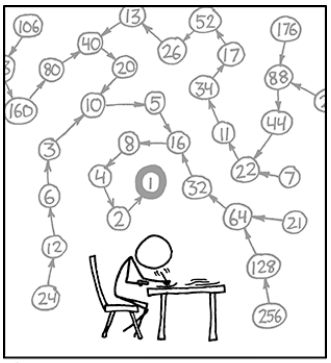
Tehnilised oskused: (15) algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek, (18) rekursiooni mälukasutus, (19) rekursiooni ajaline keerukus.

Teosta mäluga Collatzi hüpoteesi funktsioon. Algoritm peaks töötama järgnevalt [57]:

1. kui *n* on 1, siis lõpeta;
2. kui *n* on paarisarv, siis jaga 2-ga;
3. kui *n* on paaritu arv, siis korruta 3-ga ja lisa 1.

Katseta oma funktsiooni erinevate sisendite korral. Milline on funktsiooni ajaline keerukus (vt. Illustratsioon 1)?

Vihje: viimane küsimus oli trikiga küsimus. Kuna Collatzi hüpoteesi ei ole tõestatud, siis ei ole ka ajaline keerukus teada vaid võib teatud sisendite korral olla lõpmatu.



THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

Illustratsioon 1: Collatzi omapära kohta [92]

26. Listi tasandamine

Tehnilised oskused: (2) rekursiivselt lahendatud probleemi iteratiivselt lahendamine, (8) baasjuhu ja sammu äratundmine ning mõistmine, (17) fikseerimata tasemete arvuga

andmestruktuuride töötlemine, (23) rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine.

Olgu meil mitmetasemeline järjend [[1,2], 0, [[[4], 2, 4], [1]]]. Kirjuta nii iteratiivne kui ka rekursiivne funktsioon listi tasandamiseks (ingl *list flattening*) [93]. Näitena toodud järjend peaks peale tasandamist nägema välja järgnev: [1, 2, 0, 4, 2, 4, 1]. Katseta oma funktsioone muu hulgas teiste mitmetasemeliste järjenditega.

Kumba lahendust eelistad ja miks?

Vihje: listi tasandamise probleemi korral on mitterekursiivset funktsiooni keerulisem kirjutada [58].

27. Hanoi tornid

Tehnilised oskused: (5) mitmekordne rekursioon, (8) baasjuhu ja sammu äratundmine ning mõistmine, (9) alamülesandest terve ülesande lahenduseni jõudmine, (19) rekursiooni ajaline keerukus, (23) rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine.

„Hanoi tornide ülesannet tutvustas prantsuse matemaatik Edouard Lucas 1883. aastal“ [94]. Tegu on legendiga, mille päritolu ei ole teada. Iidse ennustuse kohaselt lõpeb maailm siis, kui mungad suudavad 64 ketast liigutada uuele platsile ilma ülesande põhimõtteid rikkumata. Kas on võimalik, et saame seeläbi teada maailma lõpu? [57] Uurime ja katsetame järgi!

„Nimelt on erineva läbimõõduga kettad nii laotud, et kõige suurem on kõige alumine ja väiksemad järjest üksteise peal kuni kõige väiksemani.” „Eesmärgiks on laduda kettad uuele platsile (vardale) ümber nii, et need oleksid jälle samal moel. Ühel sammul võib liigutada ainult ühte ketast ja mitte kunagi ei tohi suuremat ketast panna väiksema peale. Ümbertõstmisel saab kasutada kolmandat platsi (varrast)” [94].

Ürita kirjutada nüüd nii iteratiivset kui ka rekursiivset funktsiooni.

Vihje: Rekursiivne lahendus on tõestatult optimaalseim [57]. Hanoi tornid on selline ülesanne, mida on rekursiooniga peaaegu triviaalne, ent iteratiivset väga keeruline lahendada [19].

Kui kaua läheb aega, et liigutada kaheksa ketast? Aga millal saabub 64 ketta liigutamisega maailma lõpp?

28. Põimemeetod vs. kiirsorteerimine

Tehnilised oskused: (15) algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek, (18) rekursiooni mälukasutus, (19) rekursiooni ajaline keerukus, (23) rekursiivsete (ja iteratiivse) lahendusmeetodite vahel eelistuse tegemine.

12. ülesandes joonistasime ülalt-alla põimemeetodi rekursioonipuu. Põimemeetodit saab teostada aga kahel viisil: ülalt-alla ja alt-üles [59].

Ülalt-alla st. rekursiivne viis, mida ennist joonistasime:

1. samm: Alusta listiga „ülevalt“.
2. samm: Jaga ja sorteeri kaks järjendit rekursiivselt.
3. samm: Põimi kaks sorteeritud osa.

Alt-üles st. mitterekursiivne viis:

1. samm: Jaga järjend üleelemendilisteks järjenditeks.
2. samm: Põimi järjendid iteratiivselt alustades „alt“.

Kirjuta sorteerimisfunktsioon mõlemal viisil. Algoritmide interaktiivseid läbimänge leiad nt. järgnevatelt veebilehtedelt: http://sorting.at/ ja https://imgur.com/gallery/GD5gi.

Võrdle sama sisendi puhul ülalt-alla ja alt-üles põimemeetodite käitusaega 20. ülesandes teostatud kiirsorteerimisega. Millist meetodit/ viisi eelistaksid st. milline neist on efektiivseim?

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, _____Carola Kesküla _____,
(autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

_____Rekursiooni käsitlemine selle õpetamisel_____,
(lõputöö pealkiri)

mille juhendaja on _____Ahti Peder _____,
(juhendaja nimi)

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Carola Kesküla
07.05.2019