Kogutud materjal: Tartu, Eesti ja maailma ülikoolid

Sisukord

Tartu Ülikool

Informaatika õppekava 2018/2019 sisseastujale:
https://www.cs.ut.ee/sites/default/files/cs/infbak2018_2019.pdf

LTAT.03.001 Programmeerimine (1. semester)

- Õpiväljundid ja aine sisu:
    - https://courses.cs.ut.ee/2018/programmeerimine/fall/Main/Hindamine

    - Tunneb ja oskab kasutada: ..., rekursiooni
    - Oskab analüüsida ja üksikasjalikult selgitada programmi töö käiku ning programmi muuta, täiendada ja edasi arendada;
    - Oskab luua lihtsamale ülesandele vastava algoritmi, koostada ja korrektselt vormistada lahendusprogrammi ning seda siluda ja testida;
    - Kursus ei nõua eelteadmisi peale üldise arvutikasutusoskuse.
- Õppeviisid:
    - https://courses.cs.ut.ee/2018/programmeerimine/fall/Main/Hindamine
    - Kaks loengut, kaks praktikumi, kaks kodutööd.
- Õppematerjalid:
    - E-õpik: https://progeopik.cs.ut.ee/11_rekursioon.html
    - Projecteuler.net ülesanne: http://projecteuler.net/index.php?section=problems&id=15
    - 13. praktikum: https://courses.cs.ut.ee/2018/programmeerimine/fall/Main/Praks13
    - 14. praktikum: https://courses.cs.ut.ee/2018/programmeerimine/fall/Main/Praks14
    - 13. kodutöö: https://courses.cs.ut.ee/2018/programmeerimine/fall/Kodu13/Kodu13
    - 14. kodutöö: https://courses.cs.ut.ee/2018/programmeerimine/fall/Kodu14/Kodu14
- Esitatud ülesanded
    - Kõik õpiku ülesanded ja põhjalikud näited:
        1. [H] faktoriaal;
        2. [H] stardiloenduse modifitseerimine, et sekundeid loetaks ka pärast;
        3. [H] spiraali joonistamine kilpkonna abil;
        4. [H] Eukleidese algoritm;
        5. [H] järjendi elementide arv ilma tsükleid ja *len*-funktsiooni kasutades;
        6. [HV] Fibonacci n-is liige (x2);
        7. [HVL, HV] fraktaalide joonistamine (x2);
        8. [H] kuulujutu levik (x2);
        9. [H] küülikute ülesanne;
        10. [H] kaustanime järgi n-mõõtmelise järjendi moodustamine ning tagastamine, kus iga alamkaust on omakorda esitatud järjendina ja failid on esitatud vastavas järjendis olevate sõnadega;
        11. [H] arvamismäng;
        12. [HV] sõnast vokaalide eemaldamine;
        13. [HV] sõna tagurpidi pööramine;
        14. [HV] Projecteuler.net ülesanne (Projecteuler.net ülesanne);
        15. [HV] sugupuust eellaste leidmine;
        16. [HV] sisendjärjendi põhjal sama kujuga andmestruktuuri tagastamine, kus kõik arvud on asendatud nende absoluutväärtustega.
    - 13. praktikumi ülesanded:

1. [HV] funktsioon *esineb()*, mis tagastab tõeväärtuse vastavalt, kas väärtus esineb elementide hulgas või mitte;
2. [H] eelmise ülesande parameetriga täiendamine st. funktsioon tagastab tõeväärtuse, kas mingi väärtus esineb järjendis vähemalt n-korda (uus lisaparameeter);
3. [H] kolmeaastase lapse simulaator rekursiooniga;
4. [HV] funktsioon n-nurga joonistamiseks;
5. [HV] järjendi sügavuse arvutamine;
6. [HV] fraktaali funktsiooni kirjutamine etteantud pildi järgi (x3);

- 14. praktikumi ülesanded:
  1. [H] struktuuri sügavuse (st. mitu taset liste on maksimaalselt välimise järjendi sees) leidmine;
  2. [H] fraktaali funktsiooni kirjutamine kujundi pildi järgi;
  3. [HV] kausta sisu "failipuu" ekraanile kuvamine
  4. [HV] suurima faili leidmine etteantud kaustas (või mõnes selle alamkaustas);
  5. [H] rekursiivne funktsioon pildil oleva fraktaali joonistamiseks;
  6. [HV] labürindi läbimine.
- 13. kodutöö:
  1. [H] kasutatud auto väärtus peale n-aastat;
  2. [H] fraktaali funktsiooni kirjutamine kujundi pildi järgi;
- 14. kodutöö:
  1. [HV] järjendi tasemete korrastamine ja tagastamine;
  2. [H] sugupuu põhjal põlvnemiste ahela (eellasest järglaseni) väljastamine.

- Ülesannete klassifikatsioon tehniliste oskuste põhjal
  - iteratiivselt lahendatud probleemi rekursiooniga lahendamine:
    1. [H] e-õpiku 3. peatükist – arvamismäng;
    2. [H] kolmeaastase lapse simulaator rekursiooniga;
    3. [HV] funktsioon n-nurga joonistamiseks;
    4. [H] kasutatud auto väärtus peale n-aastat;
  - iteratiivse ja rekursiivse programmeerimise kombineerimine:
    1. [HV] järjendi sügavuse arvutamine liikudes tsükliga laiuti ja rekursiooniga süviti.
    2. [HV] 14. praktikumi 2. ülesanne – kausta sisu "failipuu" ekraanile kuvamine;
  - puurekursioon (sh binaarne rekursioon):
    1. [H] kaustapuu läbimine;
    2. [HVL, HV] fraktaalide joonistamine (x2);
    3. [H] kuulujutu levik (x2);
    4. [H] küülikute ülesanne;
    5. [H] kaustanime järgi n-mõõtmelise järjendi moodustamine ning tagastamine, kus iga alamkaust on omakorda esitatud järjendina ja failid on esitatud vastavas järjendis olevate sõnadega;

6. [HV] sugupuust eellaste leidmine;
7. [HV] kausta sisu “failipuu” ekraanile kuvamine;
8. [HV] suurima faili leidmine etteantud kaustas (või mõnes selle alamkaustas);

- lineaarne- ja sabarekursioon:
  1. [H] järjendi elementide arv ilma tsükleid ja *len*-funktsiooni kasutades;
  2. [HV] sõnast vokaalide eemaldamine;
- mõttes etteantud funktsiooni väärtustamine olenevalt sisendist:
  1. [H] faktoriaal;
- baasjuhu ja sammu äratundmine ning mõistmine:
  1. [H] faktoriaal;
  2. [HV] sõna tagurpidi pööramine;
  3. [HV] Projecteuler.net ülesanne;
  4. [HV] funktsioon *esineb()*, mis tagastab tõeväärtuse vastavalt, kas väärtus esineb elementide hulgas või mitte;
  5. [H] kasutatud auto väärtus peale n-aastat;
  6. [HV] fraktaali funktsiooni kirjutamine etteantud pildi järgi (x5);
  7. [HV] labürinti läbiva sammujärjendi leidmine;
  8. [H] sugupuu põhjal põlvnemiste ahela (eellasest järglaseni) väljastamine.
- alamülesandest terve ülesande lahenduseni jõudmine:
  1. [H] faktoriaal;
  2. [H] järjendi elementide arv ilma tsükleid ja *len*-funktsiooni kasutades;
- rekursiooni vähenemine baasjuhu suunas:
  1. [H] spiraali joonistamine kilpkonna abil;
  2. [HV] Projecteuler.net ülesanne;
  3. [HV] fraktaali funktsiooni kirjutamine kujundi pildi järgi (x3);
  4. [H] kasutatud auto väärtus peale n-aastat;
- funktsiooni käitumine ootamatute argumentide korral:
  1. [H] faktoriaal;
  2. [HV] sõna tagurpidi pööramine (tühi sõne);
  3. [HV] järjendi tasemete korrastamine ja tagastamine (sisend “[]”);
  4. [H] sugupuu põhjal põlvnemiste ahela (eellasest järglaseni) väljastamine;
- koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine:
  1. [H] stardiloenduse modifitseerimine, et sekundeid loetaks ka pärast;
  2. [HV] Fibonacci n-is liige (efektiivsem lahendus);
  3. [H] eelmise ülesande parameetriga täiendamine st. funktsioon tagastab tõeväärtuse, kas mingi väärtus esineb järjendis vähemalt n-korda (uus lisaparameeter);
- algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
  1. [H] Eukleidese algoritm;
  2. [H] küülikute ülesanne;
- fikseeritud tasemete arvuga andmestruktuuri töötlemine:

1. [HV] funktsioon *esineb()*, mis tagastab tõeväärtuse vastavalt, kas väärtus esineb elementide hulgas või mitte;

- fikseerimata tasemete arvuga andmestruktuuride töötlemine:
  1. [HV] sisendjärjendi põhjal sama kujuga andmestruktuuri tagastamine, kus kõik arvud on asendatud nende absoluutväärtustega;
  2. [HV] järjendisügavuse arvutamine;
  3. [H] struktuuri sügavuse (st. mitu taset liste on maksimaalselt välimise järjendi sees) leidmine;
  4. [HV] kausta sisu "failipuu" ekraanile kuvamine (sügavuse meelespidamine õige taande eesmärgil);
  5. [HV] suurima faili leidmine etteantud kaustas (või mõnes selle alamkaustas);
  6. [HV] järjendi tasemete korrastamine ja tagastamine;
  7. [H] sugupuu põhjal põlvnemiste ahela (eellasest järglaseni) väljastamine.

- Tähelepanekuid ja häid ideid
  - Internetiõpik on hea ja kättesaadav. Tore, et iga peatüki lõppu saab kommentaare lisada. Teeb parandusi ja küsimuste küsimist lihtsamaks.
  - Vihjed ja näidislahendused on teinekord peidetud.
  - On põhjendatud, millal rekursioon argumentideta olla võib: st. kui baas on defineeritud muutujast olenemata, siis "võib rekursioon toimuda ka väärtusi edastamata".

LTAT.03.005 Algoritmid ja andmestruktuurid (3. semester)

- Õpiväljundid ja aine sisu: https://www.is.ut.ee/rwservlet?oa_ainekava_info.rdf+1348331+PDF+0+application/pdf
  - "Oskab võrrelda algoritmide töökiirust keerukushinnangute alusel."
  - "Oskab põhilisi andmestruktuure ja algoritme arvutis realiseerida, kasutada ja sobivamaid praktikas eelistada."
  - Sisu: järjekorras 5. ja 6. loeng
    1. "loeng - Tsükkel, rekursioon, magasin, järjekord. Nendevahelised seosed.
    2. loeng - Operatsioonid magasiniga ja järjekorraga. Rekursiooni "eemaldamine""
    3. 5. ja 6. loenguslaidid kirjeldatut ei kajasta.
- Õppeviisid:
  - loengud, arvutipraktikumid, tahvlipraktikumid, tunnikontrollid loengu ajal, testid iseseisvalt, kodutööd, veebilehed kuvamaks algoritmide tööd interaktiivselt.
- Õppematerjalid:
  - Ahti Peder, Jüri Kiho, Härmel Nestra. Algoritmid ja andmestruktuurid. Ülesannete kogu (2017).

- Ahti Peder, Härmel Nestra. Algoritmid ja andmestruktuurid. Tahvlipraktikum (2018): https://moodle.ut.ee/pluginfile.php/989588/mod_resource/content/3/peafail_27082018.pdf
- I-II praktikum: https://moodle.ut.ee/pluginfile.php/1022156/mod_resource/content/1/i-ii_praks.pdf
- III-IV praktikum: https://moodle.ut.ee/pluginfile.php/1025481/mod_resource/content/1/praks3-4_oige.pdf
- IV-V praktikum: https://moodle.ut.ee/pluginfile.php/1037926/mod_resource/content/2/praks4-5.pdf
- IX-X praktikum: https://moodle.ut.ee/pluginfile.php/1060252/mod_resource/content/1/avlpraks.pdf
- XI-XII praktikum: https://moodle.ut.ee/pluginfile.php/1067210/mod_resource/content/2/aakuhik.pdf
- 3. loeng: https://moodle.ut.ee/pluginfile.php/12460/mod_resource/content/7/loeng3aasta2018.pdf
- 4. loeng: https://moodle.ut.ee/pluginfile.php/14788/mod_resource/content/8/loeng4aasta2018.pdf

- Esitatud ülesanded:
  - Kõik ülesanded:
    1. [H] arvujärjendi transponeerimine (tahvlipraktikum);
    2. [H] järjendi elementidest vähima ja suurima väärtuse leidmine (tahvlipraktikum);
    3. [HL] programm, mis väljastab ekraanile kõik n-bitised bitivektorid (tahvlipraktikum);
    4. [HV] programm, mis väljastab ekraanile kõik 30-pikkuselised bitivektorid, milles on täpselt 2 ‘1’-bitti (tahvlipraktikum);
    5. [HL] programm, mis väljastab järjendina antud hulga kõik alamhulgad (tahvlipraktikum);
    6. [H] arvujärjendi alamhulgad, mis elementide summa jääb lõiku [90, 100] (tahvlipraktikum);
    7. [H] arvujärjendi alamhulga elementide summa, mis ületab arvu 100 võimalikult vähe, olles parimal juhul sellega võrdne (tahvlipraktikum);
    8. [H] järjendi a pikim kasvav a) alamjärjend, b) osajärjend (tahvlipraktikum);
    9. [HL] mitmel eri viisil saab üles minna n-astmelisest trepist, kui iga sammuga võib võtta ühe, kaks või kolm astet (tahvlipraktikum);

10. [HL] mitmel eri viisil saab üles minna n-astmelisest trepist (väljastades kõikvõimalikud trepist ülesminekud) (x2) (tahvlipraktikum);
11. [HL] sõne permutatsioonid (tahvlipraktikum);
12. [HL] järjendist *k*-kaupa kombinatsioonid (tahvlipraktikum);
13. [H] järjendist *k*-kaupa kombinatsioonid kasutades binaarset rekursioonipuud (tahvlipraktikum);
14. [HL] rekursiivne generaator, mis annab välja *n*-bitiseid bitivektoreid (tahvlipraktikum);
15. [HV] generaator, mis annab välja hulga alamhulki (tahvlipraktikum);
16. [HL] generaatoriga sammupikkuste loetelud minemaks trepist üles (tahvlipraktikum);
17. [H] funktsiooni väärtustamine sisendi korral (x7) (ülesannetekogu);
18. [H] liitmistehete arv funktsiooni sooritamisel (ülesannetekogu);
19. [H] täisarvu n korral f(n) väärtuse arvutamisel tehtavate liitmiste ja lahutamiste arv (ülesannetekogu);
20. [H] prinditud rida arv Hanoi tornide ülesandes etteantud sisendi korral (ülesannetekogu);
21. [H] kahe sarnase funktsiooni väljakutsete arvu võrdlemine (ülesannetekogu);
22. [H] funktsioonide tühimikkohtade täitmine, et saavutada järjesitkuste naturaalarvude korrutis a-st b-ni (x5) (ülesannetekogu);
23. [H] funktsiooni poolt lahendatava ülesande mõistmine (x4) (ülesannetekogu);
24. [H] arv mitu korda vaadeldakse seljakoti ülesandel igat esemete alamhulka (x2) (ülesannetekogu);
25. [H] funktsiooniväljakutsete magasini muutumine arvutuse käigus (x4) (ülesannetekogu);
26. [H] kahe arvu korrutise funktsiooni peatumine kõigi täisarvuliste sisendpaaride korral (ülesannetekogu);
27. [H] arvu n kahega jagamisel tekkiva jäägi funktsiooni peatumine erinevate sisendite korral (x4) (ülesannetekogu);
28. [H] suvalise funktsiooni peatumine erinevate sisendite korral (x3) (ülesannetekogu);
29. [H] selgitamine, milliste sisendite korral leidub avaldisel väärtus (x4) (ülesannetekogu);
30. [H] "jaga ja valitse" tüüpi algoritmi sõnastamine, et leida järjendist suurim ja vähim element (ülesannetekogu);
31. [H] naturaalarvu kõikvõimalikud lahutused arvude [n] summadeks (x2) (ülesannetekogu);
32. [H] võimalused, millised vahesummade jadad saavad tekkida pliiatsite loendamisel, kui karbis on täpselt n pliiatsit (ülesannetekogu);
33. [H] permutatsioonide ülesande lahendamine neljal erineval klassikalisel järjestamismeetodil (valikumeetod, pistemeetod, kiirmeetod, põimemeetod) (ülesannetekogu);

34. [H] suurim väärtus, mille funktsioon suudab antud ajalise piirangu jooksul genereerida (x4) (I-II praktikum);
35. [H] meetodi tööaeg, kus Fibonacci on realiseeritud mitterekursiivse algoritmiga (I-II praktikum);
36. [H] rekursiivse ja mitterekursiivse Fibonacci funktsiooni võimsuse võrdlemine (I-II praktikum);
37. [H] ekraanile bikivektorite (pikkusega n) väljastamine (III-IV praktikum);
38. [H] ekraanile bitivektorite (pikkusega n ja milles on parajasti k > 0 ühte) väljastamine (III-IV praktikum);
39. [H] naturaalarvu kõikvõimalikud lahutused liidetavate 1 ja 2 summadeks (III-IV praktikum);
40. [H] täisarvu järjendi kõikvõimalike alamhulkade summad (III-IV praktikum);
41. [H] järjendi kõik elementide kombinatsioonid etteantud arvu k > 0 kaupa (III-IV praktikum);
42. [H] kiirsorteerimise meetod masiivil viisil, kus rekursiivseid väljakutseid töötlemata osamassiive peab haldama magasini abil (IV-V praktikum);
43. [H] juhusliku AVL-puu kõrgusega h koostamine (IX-X praktikum);
44. [H] rekursiivne meetod arvude järjendi kuhjastamiseks (XI-XII praktikum);

- Ülesannete klassifikatsioon tehniliste oskuste põhjal:
  - rekursiivselt lahendatud probleemi iteratiivselt lahendamine:
    1. [H] meetodi tööaeg, kus Fibonacci on realiseeritud mitterekursiivse algoritmiga;
  - puurekursioon (sh binaarne rekursioon):
    1. [HL] järjendi summa arvutamine;
    2. [H] järjendi elementidest vähima ja suurima väärtuse leidmine (isejoonistatud rekursiooni skeemi põhjal);
    3. [HL] programm, mis väljastab ekraanile kõik n-bitised bitivektorid;
    4. [HV] programm, mis väljastab ekraanile kõik 30-pikkuselised bitivektorid, milles on täpselt 2 ‘1’-bitti;
    5. [HL] programm, mis väljastab järjendina antud hulga kõik alamhulgad;
    6. [H] arvujärjendi alamhulgad, mis elementide summa jääb lõiku [90, 100];
    7. [H] arvujärjendi alamhulga elementide summa, mis ületab arvu 100 võimalikult vähe, olles parimal juhul sellega võrdne;
    8. [H] järjendi a pikim kasvav a) alamjärjend, b) osajärjend;
    9. [HV] täisarvude järjendi suurima elementide arvuga (ühe) osajärjendi leidmine, mille summa on 0;
    10. [H] juhusliku AVL-puu kõrgusega h koostamine;
    11. [H] rekursiivne meetod arvude järjendi kuhjastamiseks;
  - lineaarne- ja sabarekursioon:

1. [HL] järjendi summa arvutamine;
2. [H] järjendi elementidest vähima ja suurima väärtuse leidmine (isejoonistatud rekursiooni skeemi põhjal);

- mitmekordne rekursioon:
    1. [HL] mitmel eri viisil saab üles minna n-astmelisest trepist, kui iga sammuga võib võtta ühe, kaks või kolm astet;
    2. [HL] mitmel eri viisil saab üles minna n-astmelisest trepist (väljastades kõikvõimalikud trepist ülesminekud);
    3. [HL] sõne permutatsioonid;
    4. [HL] järjendist k kaupa kombinatsioonid;
    5. [H] järjendist *k*-kaupa kombinatsioonid kasutades binaarset rekursioonipuud;
    6. [HL] rekursiivne generaator, mis annab välja *n*-bitiseid bitivektoreid;
    7. [HL] rekursiivne generaator, mis annab välja hulga alamhulki, mille summa jääb antud lõiku;
- mõttes etteantud funktsiooni väärtustamine olenevalt sisendist:
    1. [H] funktsiooni väärtustamine sisendi korral (x7);
    2. [H] prinditud rida arv Hanoi tornide ülesandes etteantud sisendi korral;
    3. [H] kahe sarnase funktsiooni väljakutsete arvu võrdlemine;
- baasjuhu ja sammu äratundmine ning mõistmine:
    1. [HL] mitmel eri viisil saab üles minna n-astmelisest trepist, kui iga sammuga võib võtta ühe, kaks või kolm astet;
    2. [HL] mitmel eri viisil saab üles minna n-astmelisest trepist (väljastades kõikvõimalikud trepist ülesminekud);
    3. [H] funktsiooni poolt lahendatava ülesande mõistmine (x4);
    4. [H] “jaga ja valitse” tüüpi algoritmi sõnastamine, et leida järjendist suurim ja vähim element;
    5. [H] naturaalarvu kõikvõimalikud lahutused arvude [n] summadeks (x2);
    6. [H] võimalused, millised vahesummade jadad saavad tekkida pliiatsite loendamisel, kui karbis on täpselt n pliiatsit;
    7. [H] permutatsioonide ülesande lahendamine neljal erineval klassikalisel järjestamismeetodil (valikumeetod, pistemeetod, kiirmeetod, põimemeetod);
    8. [H] ekraanile bikivektorite (pikkusega n) väljastamine;
    9. [H] ekraanile bitivektorite (pikkusega n ja milles on parajasti k > 0 ühte) väljastamine;
    10. [H] kiirsorteerimise meetod masiivil viisil, kus rekursiivseid väljakutseid töötlemata osamassiive peab haldama magasini abil;
    11. [H] rekursiivne meetod arvude järjendi kuhjastamiseks;
- alamülesandest terve ülesande lahenduseni jõudmine:
    1. [H] “jaga ja valitse” tüüpi algoritmi sõnastamine, et leida järjendist suurim ja vähim element;
    2. [H] ekraanile bikivektorite (pikkusega n) väljastamine;

3. [H] ekraanile bitivektorite (pikkusega n ja milles on parajasti k > 0 ühte) väljastamine;
4. [H] naturaalarvu kõikvõimalikud lahutused liidetavate 1 ja 2 summadeks;
5. [H] täisarvu järjendi kõikvõimalike alamhulkade summad;
6. [H] järjendi kõik elementide kombinatsioonid etteantud arvu k > 0 kaupa;

- rekursiooni vähenemine baasjuhu suunas:
  1. [H] liitmistehete arv funktsiooni sooritamisel;
  2. [H] täisarvu n korral f(n) väärtuse arvutamisel tehtavate liitmiste ja lahutamiste arv;
  3. [H] kahe arvu korrutise funktsiooni peatumine kõigi täisarvuliste sisendpaaride korral;
  4. [H] arvu n kahega jagamisel tekkiva jäägi funktsiooni peatumine erinevate sisendite puhul (x4);
  5. [H] suvalise funktsiooni peatumine erinevate sisendite korral (x3);
  6. [H] "jaga ja valitse" tüüpi algoritmi sõnastamine, et leida järjendist suurim ja vähim element;
- funktsiooni käitumine ootamatute argumentide korral:
  1. [H] kahe arvu korrutise funktsiooni peatumine kõigi täisarvuliste sisendpaaride korral;
  2. [H] arvu n kahega jagamisel tekkiva jäägi funktsiooni peatumine erinevate sisendite puhul (x4);
  3. [H] suvalise funktsiooni peatumine erinevate sisendite korral (x3);
  4. [H] selgitamine, milliste sisendite korral leidub avaldisel väärtus (x4);
  5. [H] ekraanile bitivektorite (pikkusega n ja milles on parajasti k > 0 ühte) väljastamine (suure argumendi korral);
- rekursiivse funktsiooni olemus (nt. rekursiivne funktsioon ei pea alati midagi tagastama ega edastama):
  - [H] funktsiooni poolt lahendatava ülesande mõistmine (x4);
- rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast:
  - [H] kahe sarnase funktsiooni väljakutsete arvu võrdlemine;
  - [H] arv mitu korda vaadeldakse seljakoti ülesandel igat esemete alamhulka (x2);
- koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine:
  - [H] arvujärjendi transponeerimine (meetodi sisu kompaktsemalt kirjutamine);
  - [HV] programm, mis väljastab ekraanile kõik 30-pikkuselised bitivektorid, milles on täpselt 2 '1'-bitti;
  - [H] mitmel eri viisil saab üles minna n-astmelisest trepist (väljastades kõikvõimalikud trepist ülesminekud) (tagastada kõikvõimalikud ülesminekud järjendina kasutades vaid kahte baasjuhtu);
  - [HL] generaatoriga sammupikkuste loetelud minemaks trepist üles;

        - [H] funktsioonide tühimikkohtade täitmine, et saavutada järjesitkuste naturaalarvude korrutis a-st b-ni (x5);
        - [H] ekraanile bitivektorite (pikkusega n ja milles on parajasti k > 0 ühte) väljastamine;
        - [H] kiirsorteerimise meetod massiivil viisil, kus rekursiivseid väljakutseid töötlemata osamassiive peab haldama magasini abil;
    - algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
        - [HL] järjendi summa arvutamine;
        - [HL] järjendist k kaupa kombinatsioonid;
        - [H] juhusliku AVL-puu kõrgusega h koostamine;
    - rekursiooni mälukasutus:
        - [H] ekraanile bitivektorite (pikkusega n ja milles on parajasti k > 0 ühte) väljastamine;
    - rekursiooni ajaline keerukus;
        - [H] suurim väärtus, mille funktsioon suudab antud ajalise piirangu jooksul genereerida (x4);
        - [H] ekraanile bitivektorite (pikkusega n ja milles on parajasti k > 0 ühte) väljastamine;
    - iteratiivse ja rekursiivse koodi ajalise keerukuse vahe:
        - [H] rekursiivse ja mitterekursiivse Fibonacci funktsiooni võimsuse võrdlemine;
    - aktiveerimiskirjete ja/või rekursioonipuu joonistamine funktsiooni väljakutse põhjal:
        - [H] järjendi elementidest vähima ja suurima väärtuse leidmine;
        - [H] järjendi minimaalse ja maksimaalse elemendi leidmine (unaarse ja binaarse rekursiooni skeem);
        - [HL] programm, mis väljastab ekraanile kõik n-bitised bitivektorid;
        - [HL] programm, mis väljastab järjendina antud hulga kõik alamhulgad;
        - [HL] mitmel eri viisil saab üles minna n-astmelisest trepist, kui iga sammuga võib võtta ühe, kaks või kolm astet;
        - [HL] sõne permutatsioonid;
        - [H] järjendist *k*-kaupa kombinatsioonid kasutades binaarset rekursioonipuud;
        - [H] funktsiooniväljakutsete magasini muutumine arvutuse käigus (x4);
        - [H] ekraanile bikivektorite (pikkusega n) väljastamine;
        - [H] ekraanile bitivektorite (pikkusega n ja milles on parajasti k > 0 ühte) väljastamine;
        - [H] naturaalarvu kõikvõimalikud lahutused liidetavate 1 ja 2 summadeks;
        - [H] täisarvu järjendi kõikvõimalike alamhulkade summad;
        - [H] järjendi kõik elementide kombinatsioonid etteantud arvu k > 0 kaupa;
    - rekursiivsed generaatorid:
        - [HL] rekursiivne generaator, mis annab välja *n*-bitiseid bitivektoreid;

        - [HV] generaator, mis annab välja hulga alamhulki;
        - [HL] rekursiivne generaator, mis annab välja hulga alamhulki, mille summa jääb antud lõiku;
        - [HV] täisarvude järjendi suurima elementide arvuga (ühe) osajärjendi leidmine, mille summa on 0;
        - [HL] generaatoriga sammupikkuste loetelud minemaks trepist üles;
- Tähelepanekuid ja häid ideid
    - "Rekursiooni disain ja koostamine" (3. loeng):
        - "Leia kuidas ülesannet lihtsamateks alamülesanneteks jagada
        - Leia kuidas tuletada alamülesannete lahendustest käesoleva ülesande lahendus
        - Leia lõpetamistingimus mille poole jagamine viib
        - Määra lahendus kui lõpetamistingimus on täidetud
        - Aga ...., tihti on see keeruline"
        - Mõeldakse rekursiooni disainist kui puu osadest (juur st. lähtekoht, lehed st. osad, mida enam ei jaota, vahetipud st. rekursiooni uus väljakutse ja served st. alamprogrammide väljakutsete ja tagastuste analüüs).
        - Mainitakse tagasipöördumistehnikat ehk tagurdamist (backtracking)
    - 4. loeng:
        - Põhiliseks lahendusmeetodiks on hargnevuste puu konstrueerimine ja sellel liikumise määramine;
        - Mainitakse lipud malelaual (n-queen probleem) probleemi;
        - Rekursioon ei sobi, kui tahame midagi teha "lõpmatult" palju.
    - Tahvlipraktikumid failist:
        - Rekursiooni mõiste käiakse ühe lausega uuesti üle, aga väga teistsuguses võtmes, kui eelnevas programmeerimise aines ("Rekursioon on programmeerimistehnika, milles sisendparameetri(te)ga määratud ülesanne taandatakse samasugus(t)ele, aga teise (üldjuhul väiksema) parameetri(te) väärtusega ülesandele. Võimalik on ka olukord, kus rekursiivsel algoritmil (või funktsioonil) parameetreid pole, kuid selliseid käesoleval kursusel ei käsitleta.")
        - Mis on oluline rekursiooni juures?
            1. "Oluline on koostatava, antud ülesannet lahendava funktsiooni nn väline spetsifitseerimine: tuleb täpselt iseloomustada parameetrite otstarve ja samuti kirjeldada oodatav tulemus."
        - Ülesannete juures on toodud välja asjakohased mõisted. Väga hea mõte!
        - Käiakse üle mõisted unaarne (ehk lineaarne) rekursioon ja sabarekursioon.
        - Tuuakse sisse uus mõiste rekursioonipuu ja kirjeldatakse seda.
        - Rekursiivset funktsiooni saab alati iteratiivseks teha?
            2. "Kuid binaarse rekursiooni juhul, kus funktsiooni või protseduuri kehas on ette nähtud rekursiivne väljakutse kahel korral, ei pruugi (loomulikku) iteratiivset analoogi alati leidudagi."
        - Millal kasutada binaarset rekursiooni?

3. “Binaarse rekursiivse funktsiooni või protseduuri koostamise vajadus võib otseselt tuleneda nii ülesande püstitusest (nagu nt Fibonacci arvude juhul) kui ka asjaolust, et muul viisil (nt iteratiivsel moel) on antud ülesande lahendamine oluliselt keerukam.”

- Kuidas konstrueerida etteantud ülesannet lahendavat rekursiivset programmi?
  4. “Seejuures koostame skeemi, mis väljendab „tavaelule sarnanevat“ kõikide variantide süsteemset ülevaatust nende töötlemise järjekorras. Põhieesmärgiks on saavutada olukord, kus skeemina esitatud võimalused jagunevad ühetaolisel viisil skeemi igas punktis; st skeemiks on tegelikult rekursioonipuu. Seejärel proovime koostada seda variantide osadeks jaotavat tööprotsessi (rekursioonipuud) järgivat rekursiivset programmi.”
- Multirekursiooni kohta, et seda ei ole mõistlik praktilises programmeerimistöös kasutada:
  5. “Järgnevates näidetetes vaatleme permutatsioone ja kombinatsioone väljastavate multirekursiivsete protseduuride disainimist. Nendime, et tegemist on õppeotstarbelise suunitlusega aruteludega. Praktilises programmeerimistöös on mõistlik kasutada vastavaid permutatsioonide ja kombinatsioonide tarkvaralisi generaatoreid, Pythonis näiteks teegis itertools leiduvaid vahendeid.”
- Üritatakse õpetada erinevaid jaotamisprintsiipe.
- Rekursiivsed generaatorid. Tutvustatakse mõistet.
  6. Miks? “Praktikas on märksa suurema tähtsusega protseduurid, mis väljastamise (printimise) asemel genereerivad ehk annavad välja (yield) leitud väärtusi.”
  7. “Lugejalt eeldame Python-põhise generaatori mõiste tundmist.”
  8. Ülesande 3.17 lahenduses kaks viga. Viimasest *for*-loopist puudu koolon ja järjendisse lisamisel peaks (n, tee+’0’) asemel olema (n, tee+järjend(‘0’))

- Liiga palju ülesandeid! Konkreetselt õpitavad oskused peaksid ülesande juures kirjas olema. Muidu ei oska tudeng valida olulisemaid ja valib tõenäoliselt ajanappuses mõned suvalised.

LTAT.03.006 Automaadid, keeled ja translaatorid (4. semester)

- Õpiväljundid ja aine sisu https://www.is.ut.ee/rwservlet?oa_ainekava_info.rdf+1349795+PDF+0+application/pdf:
  - Rekursiooni ei mainita õpiväljundites.
  - 6. loeng ajakavas kirjeldusega: “Süntaksanalüüs: ülalt-alla parsimine, vasakrekursiooni elimineerimine, vasakfaktoriseerimine.”
- Õppeviisid:
  - loengud, loengutestid, praktikumid, kodutööd, kordamistestid.

- Õppematerjalid (mõlemad märgitud nii kohustuslike kui ka soovituslike õppematerjalide nimistus):
  - *Introduction to Compiler Design;*
  - *Compiler Design: Virtual Machines.*
  - 5. praktikum: https://courses.cs.ut.ee/2019/AKT/Main/Praks5
  - 7. praktikum: https://courses.cs.ut.ee/2019/AKT/Main/Praks7
  - Tervitav materjal: https://courses.cs.ut.ee/2019/AKT/Main/Syllabus
- Esitatud ülesanded
  - Kõik esitatud ülesanded:
    1. "4. kodutöö: Lõpliku automaadi realiseerimine."
       1. [H] rekursiivse lahendus, kus vältimaks *StackOverflowException*'it või *OutOfMemoryError*'it ei tohi rekursiooni sügavus sõltuda sisendi pikkusest.
    2. "6. Grammatika ja lekser" – "Kontekstivaba grammatika"
       1. [H] rekursiivne algoritm, mis väljastab regulaaravaldisega samaväärse grammatika.
    3. "7. Käsitsi parsimine"
       1. [H] keele a^nb^n jaoks parseri kirjutamine, mis tunneks neid sõnu, mis sinna keelde kuuluvad;
       2. [H] kalalekserile jätkuks ka kalaparser;
       3. [HV] grammatikast vasakrekursiooni elimineerimine.
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal:
    1. baasjuhu ja sammu äratundmine ning mõistmine:
       1. [H] rekursiivse lahendus, kus vältimaks *StackOverflowException*'it või *OutOfMemoryError*'it ei tohi rekursiooni sügavus sõltuda sisendi pikkusest;
       2. [H] rekursiivne algoritm, mis väljastab regulaaravaldisega samaväärse grammatika;
       3. [H] keele a^nb^n jaoks parseri kirjutamine, mis tunneks neid sõnu, mis sinna keelde kuuluvad;
    2. rekursiooni vähenemine baasjuhu suunas:
       1. [H] rekursiivse lahendus, kus vältimaks *StackOverflowException*'it või *OutOfMemoryError*'it ei tohi rekursiooni sügavus sõltuda sisendi pikkusest.
       2. [H] rekursiivne algoritm, mis väljastab regulaaravaldisega samaväärse grammatika;
       3. [H] keele a^nb^n jaoks parseri kirjutamine, mis tunneks neid sõnu, mis sinna keelde kuuluvad;
    3. rekursiivse funktsiooni olemus (nt. rekursiivne funktsioon ei pea alati midagi tagastama ega edastama):
       1. [HV] grammatikast vasakrekursiooni elimineerimine.
    4. rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast:

            1. [H] rekursiivse lahendus, kus vältimaks *StackOverflowException*'it või *OutOfMemoryError*'it ei tohi rekursiooni sügavus sõltuda sisendi pikkusest.
        5. fikseerimata tasemete arvuga andmestruktuuride töötlemine:
            1. [H] rekursiivse lahendus, kus vältimaks *StackOverflowException*'it või *OutOfMemoryError*'it ei tohi rekursiooni sügavus sõltuda sisendi pikkusest.
- Tähelepanekuid ja häid ideid
    - Rekursiooni mõistet ja põhimõtteid ei käida üle.
    - "7. Parser. Parseri kirjutamine on oluliselt mõnusam ja tegelikult üks suurepärane harjutus rekursiooni kohta." (tervitav materjal);
    - 5. praktikum "Avaldise struktuur" "Puu töötlemine rekursiooniga ei ole ainult niisama harjutus, vaid selles peitub kogu Tõde programmeerimise kohta!" (5. praktikum);
    - "7. Käsitsi parsimine": "Kuna need teemad muutuvad üsna tehnilisteks, tuleb meeles pidada, et eesmärk on siin arendada modulaarset mõtteviisi. Grammatika ja rekursioon, nendes juba peituvad kogu informaatika põhitõed, aga kõrgema taseme saavutamiseks on eriti kasulik üritada ise välja mõelda, kuidas vasakrekursiooni elimineerimise järel peab tagastama esialgse (vasakassotsiatiivse operaatorite puhul) parsepuu. Kui saad sellest teemast aru, siis võib Sind turvalise tarkvara arendamisega usaldada!" (7. praktikum).

Eesti ülikoolid

Tallinna Tehnikaülikool ehk TalTech

Informaatika õppekava IAIB17/19:
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=

ITI0102 Programmeerimise algkursus (1. semester)

- Õpiväljundid ja aine sisu https://ois.ttu.ee/portal/page?_pageid=37,674581&_dad=portal&_schema=PORTAL&p_action=view&p_id=115197&p_session_id=72910354&p_public=1&p_mode=1&keel=ET:
    1. Rekursiooni ei mainita.
- Õppeviisid https://courses.cs.ttu.ee/pages/ITI0102:
    1. loengud, praktikumid, harjutused, tunnikontroll
    2. keel: Python
- Õppematerjalid:

1. Õpik: John M. Zelle. Python Programming: An Introduction to Computer Science. 3rd ed. 2016
2. E-õpik: https://ained.ttu.ee/pydoc/
3. Aine koduleht: https://courses.cs.ttu.ee/pages/ITI0102
4. Loenguslaidid: https://ained.ttu.ee/pydoc/lectures/loeng_recursion.pptx
5. 1. loeng: https://gitpitch.com/ttupy/slides

- Esitatud ülesanded ja näited:
  - Kõik ülesanded:
    1. [N/HV] rekursiivne otsing – alamsõne indeks sõnes (e-õpik);
    2. [H] Fibonacci modifitseerimine, et see prindiks välja sõnumi, kui funktsiooni kutsutakse ja kui see midagi tagastab (õpik);
    3. [HV] Fibonacci, mis loendab mitu korda funktsiooni välja kutsutakse fib(n) korral, kus n on kasutajasisend (õpik);
    4. [HV] palindroom (õpik);
    5. [H] järjendi suurim number (õpik);
    6. [HV] teistsuguse baasiga numbrisüsteemi uued arvud st. sisendnumbri ja baasi põhjal sisendnumbri uued arvud (õpik);
    7. [HV] numbrite sõnaline väljastamine (õpik);
    8. [HV] C(n,k) kombinatsioonid (õpik);
    9. [H] fraktaal – Kochi lumehelbe funktsioon pildi järgi (õpik);
    10. [HV] fraktaal – C-kurvi funktsioon pildi järgi (õpik);
    11. [H] tekstist vigaste ja tundmatute sõnade tuvastamine (õpik);
    12. [H] segamini sõnasisendi anagrammide loomine ja sõnastikust vaste otsimine (õpik);
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal
    - koodikorduse vältimine;
      1. [N/HV] rekursiivne otsing – alamsõne indeks sõnes;
      2. [H] fraktaal – Kochi lumehelbe funktsioon pildi järgi;
      3. [HV] fraktaal – C-kurvi funktsioon pildi järgi;
    - puurekursioon (sh binaarne rekursioon):
      1. [H] tekstist vigaste ja tundmatute sõnade tuvastamine;
    - baasjuhu ja sammu äratundmine ning mõistmine:
      1. [N/HV] rekursiivne otsing – alamsõne indeks sõnes;
      2. [H] palindroom;
      3. [H] järjendi suurim number;
      4. [HV] teistsuguse baasiga numbrisüsteemi uued arvud;
      5. [HV] numbrite sõnaline väljastamine;
      6. [H] fraktaal – Kochi lumehelbe funktsioon pildi järgi;
      7. [HV] fraktaal – C-kurvi funktsioon pildi järgi;
      8. [H] segamini sõnasisendi anagrammide loomine ja sõnastikust vaste otsimine;
    - alamülesandest terve ülesande lahenduseni jõudmine:
      1. [H] fraktaal – Kochi lumehelbe funktsioon pildi järgi;
      2. [HV] fraktaal – C-kurvi funktsioon pildi järgi;

        - rekursiooni vähenemine baasjuhu suunas:
            1. [HV] teistsuguse baasiga numbrisüsteemi uued arvud;
        - funktsiooni käitumine ootamatute argumentide korral:
            1. [HV] palindroom;
        - rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast:
            1. [N/HV] rekursiivne otsing – alamsõne indeks sõnes;
            2. [H] Fibonacci modifitseerimine, et see prindiks välja sõnumi, kui funktsiooni kutsutakse ja kui see midagi tagastab;
            3. [HV] Fibonacci, mis loendab mitu korda funktsiooni välja kutsutakse;
        - koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine:
            1. [H] Fibonacci modifitseerimine, et see prindiks välja sõnumi, kui funktsiooni kutsutakse ja kui see midagi tagastab;
            2. [HV] Fibonacci, mis loendab mitu korda funktsiooni välja kutsutakse;
        - algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
            1. [H] C(n,k) kombinatsioonid;
        - iteratiivse ja rekursiivse koodi ajalise keerukuse vahe:
            1. [HV] C(n,k) kombinatsioonid;
- Tähelepanekuid ja häid ideid
    1. Rekursiooni õpetatakse 9 nädalal (kokku 16 nädalat). Peale "failid (lugemine, kirjutamine), sortimine) ja enne kontrolltööd ning "objektid, klassid" teemat. (1. loeng)
    2. Sabarekursiooni kohta: "Selline kirjaviis võimaldab rekursiooni optimeerida sama kiireks kui iteratiivset lahendust
        1. aga mitte tavalises Pythonis!
        2. on täiendavaid mooduleid, mis seda võimaldavad" (loenguslaidid);
    3. Väidavad, et: "Pythonis on rekursiivne lahendus alati ebaefektiivsem kui analoogne iteratiivne lahendus" (loenguslaidid);
    4. Alustatakse matemaatikast: "Matemaatikas kasutatakse rekutsiooni rekurrentsetes arvujadades. Arvujada on rekurrentne siis, kui jada mingi väärtus sõltub eelmisest väärtusest." (e-õpik);
    5. Näidete 'faktoriaal' ja 'rekursiivne otsing – alamsõne indeks sõnes' lahendusmeetodid tehti aeglaselt, põhjalikult ja selgitatult läbi.
    6. Võrreldi kahte sama eesmärgiga Fibonacci rekursiivset ja iteratiivset lahendusmeetodit. Täiendati rekursiivset iteratiivse heade külgede põhjal, et ta oleks mäluga ja seega iteratiivsest kiirem. Tehti kiirusevõrdlusi.
    7. Lahendusmeetod: "enne proovime määrata tingimusi, kui rekursiooni ei pea kasutama." Lihtsad punktid pandi kohe koodi kirja. (e-õpik);

8. Lahendusmeetod ülesande [N/HV] rekursiivne otsing – alamsõne indeks sõnes juures:
   "Selles ja sarnastes ülesannetes võiks kasutada sellist algoritmi:
   1. Jaga sõne 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." (e-õpik)
9. Täiesti lahenduseta ülesandeid ei olnud!
10. Baasi ja rekursiivse(te) juhtude defineerimine: "The simplest way to guarantee that these two conditions are met is to make sure that each recursion always occurs on a smaller version of the original problem. A very small version of the problem that can be solved without recursion then becomes the base case." (õpik)
11. Rekursioon vs. iteratsioon (õpik):
    - "In fact, recursive functions are a generalization of loops. Anything that can be done with a loop can also be done by a simple kind of recursive function. In fact, there are programming languages that use recursion exclusively. On the other hand, some things that can be done very simply using recursion are quite difficult to do with loops."
    - "But you have to be careful. It's also possible to write some very inefficient recursive algorithms. One classic example is calculating the nth Fibonacci number."
    - "Recursion is closely related to mathematical induction, and it requires practice before it becomes comfortable. As long as you follow the rules and make sure that every recursive chain of calls eventually reaches a base case, your algorithms will work."
    - Hanoi tornide ülesandes rekursiooni kasutamine teeb ülesande triviaalseks: "Sometimes using recursion can make otherwise difficult problems almost trivial."

ITI0202 Programmeerimise põhikursus (2. semester)

- Õpiväljundid ja aine sisu https://ois.ttu.ee/portal/page?_pageid=37,674581&_dad=portal&_schema=PORTAL&p_action=view&p_id=120737&p_session_id=72910354&p_public=1&p_mode=1&keel=ET:
  1. Rekursiooni ei mainita.
- Õppeviisid:
  1. loengud, praktikumid, projektipraktikumid, konsultatsioonid
  2. keel: Java
- Õppematerjalid:
  1. Õpik: Introduction to Java Programming (Brief Version) Y. Daniel Liang (2015)

2. Kursuse veebileht: https://courses.cs.ttu.ee/pages/ITI0202
3. E-õpik: https://ained.ttu.ee/javadoc/rekursioon.html
4. 12. ülesanne (rekursiivne otsing): https://ained.ttu.ee/javadoc/solutions/EX12.html

- Esitatud ülesanded
  - Kõik ülesanded:
    1. [HL] rekursiivne otsing – alamsõne indeks sõnes;
    2. Õpikust:
       1. [H] faktoriaali funktsiooni väljakutsete arv;
       2. [H] tundmatu rekursiivse funktsiooni väljund, baas ja samm;
       3. [H] rekursiivse matemaatilise definitsiooni kirjutamine (x3);
       4. [H] palindroom (x4);
       5. [H] selection sort;
       6. [H] kausta suurus (x5);
       7. [H, HV] Hanoi tornid (x2);
       8. [H] fraktaal – Sierpniski kolmnurk (x3);
       9. [H] faktoriaal rekursiooniga;
       10. [H] Fibonacci iteratiivselt;
       11. [H] suurim ühistegur;
       12. [H] jada summa (x3);
       13. [HV] Fibonacci väljakutsete arv;
       14. [H] arv ümberpööratuna;
       15. [H] sõne ümberpööratuna (x2);
       16. [H] tähe esinemise arv sõnes (x2);
       17. [H] täisarvu numbrite summa;
       18. [H] suurim arv massiivis;
       19. [H] sõnes esinevate suurtähtede arv;
       20. [H] massiivis esinevate suurtähtede arv;
       21. [H] tähe esinemise arv massiivis;
       22. [H] fraktaal – ringide kuvamine;
       23. [H] kümnendarv binaararvuks;
       24. [H] kümnendarv kuueteistkümnendarvuks;
       25. [H] binaararv kümnendarvuks;
       26. [H] kuueteistkümnendarv kümnendarvuks;
       27. [HV] sõne permutatsioonid;
       28. [H] tee leidmine labürindis;
       29. [H] kaustas olevate failide arv;
       30. [H] sõna esinemise arv etteantud kausta kõikides failides;
       31. [H] sõna asendamine etteantud kausta kõikides failides;
       32. [H] mäng: rüütli ringreis (ingl *Knight's Tour*);
       33. [H] lipud malelaual (n-queens);
       34. [H] fraktaal – H-puu;

35. [H] fraktaal – Hilberti kurv;
36. [H] fraktaal – rekursiivne puu;

- Ülesannete klassifikatsioon tehniliste oskuste põhjal
  - iteratiivselt lahendatud probleemi rekursiooniga lahendamine:
    1. [H] faktoriaal rekursiooniga;
  - rekursiivselt lahendatud probleemi iteratiivselt lahendamine:
    1. [H] Fibonacci iteratiivselt;
  - puurekursioon (sh binaarne rekursioon):
    1. [H] kaustas olevate failide arv;
    2. [H] sõna esinemise arv etteantud kausta kõikides failides;
    3. [H] sõna asendamine etteantud kausta kõikides failides;
  - lineaarne- ja sabarekursioon:
    1. [HV] sõne permutatsioonid (abifunktsioon);
    2. [H] sõne ümberpööratuna (abifunktsioon);
    3. [H] tähe esinemise arv sõnes (abifunktsioon);
    4. [H] massiivis esinevate suurtähtede arv (abifunktsioon);
    5. [H] tähe esinemise arv massiivis (abifunktsioon);
  - mõttes etteantud funktsiooni järgi avaldise väärtustamine olenevalt sisendist:
    1. [H] tundmatu rekursiivse funktsiooni väljund, baas ja samm;
    2. [H] kausta suurus;
    3. [H] Hanoi tornid;
  - baasjuhu ja sammu äratundmine ning mõistmine:
    1. [HL] rekursiivne otsing – alamsõne indeks sõnes;
    2. [H] tundmatu rekursiivse funktsiooni väljund, baas ja samm;
    3. [H] rekursiivse matemaatilise definitsiooni kirjutamine (x3);
    4. [H] palindroomi funktsioon baas;
    5. [H] kausta suurus;
    6. [H] fraktaal – Sierpniski kolmnurk (x3);
    7. [H] suurim ühistegur;
    8. [H] arv ümberpööratuna;
    9. [H] sõne ümberpööratuna;
    10. [H] suurim arv massiivis;
    11. [H] sõnes esinevate suurtähtede arv;
    12. [H] massiivis esinevate suurtähtede arv;
    13. [H] tähe esinemise arv massiivis;
    14. [H] kümnendarv binaararvuks;
    15. [H] kümnendarv kuueteistkümnendarvuks;
    16. [H] binaararv kümnendarvuks;
    17. [H] kuueteistkümnendarv kümnendarvuks;
    18. [HV] sõne permutatsioonid;
    19. [H] tee leidmine labürindis;
    20. [H] mäng: rüütli ringreis (ingl *Knight's Tour*);
    21. [H] lipud malelaual (n-queens);

22. [H] fraktaal – Hilberti kurv;
23. [H] fraktaal – rekursiivne puu;

- rekursiooni vähenemine baasjuhu suunas:
  1. [H] tähe esinemise arv sõnes;
  2. [H] täisarvu numbrite summa;
  3. [H] fraktaal – ringide kuvamine;
- funktsiooni käitumine ootamatute argumentide korral:
  1. [H] kausta suurus (tühi kaust);
  2. [H] fraktaal – Sierpniski kolmnurk (negatiivse taseme korral);
- rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast:
  1. [HL] rekursiivne otsing – alamsõne indeks sõnes;
  2. [H] faktoriaali funktsiooni väljakutsete arv;
  3. [H] palindroomi funktsiooni väljakutsete arv etteantud sisendi korral;
  4. [H] fraktaal – Sierpniski kolmnurk (funktsiooni väljakutsete arv tasemeti);
  5. [HV] Fibonacci väljakutsete arv;
- koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine:
  1. [H] kausta suurus (kas programm töötab koodirea asenduse korral?) (x2);
  2. [H] fraktaal – Sierpniski kolmnurk (x2);
  3. [HV] Fibonacci väljakutsete arv;
  4. [H] sõne ümberpööratuna;
  5. [H] tähe esinemise arv sõnes;
  6. [HV] Hanoi tornid;
  7. [H] kausta suurus (rekursioonita);
- algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
  1. [H] jada summa (x3);
  2. [H] fraktaal – H-puu;
- aktiveerimiskirjete ja/või rekursioonipuu koostamine;
  1. [H] palindroomi funktsiooni väljakutsete kuhi (x2);
  2. [H] selection sort väljakutsete kuhi;
- rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine:
  1. [H] kausta suurus (rekursioonita st. keerukam);

- Tähelepanekuid ja häid ideid
  1. E-õpikus käiakse rekursiooni olemus ja mõiste uuesti üle. Väga hea!
  2. E-õpikus ainult üks ülesanne.

3. “Java keel kahjuks ei oska rekursiooniga optimaalselt hakkama saada. Seega, rekursiivne väljakutse on tavaliselt natuke aeglasem kui iteratiivne. See vahe on minimaalne (või olematu). Pigem valitakse rekursiivne lahendus seetõttu, et see on lühem ja selgem.”
4. Väga kasulik, et selle ja ka eelneval ainel on olemas e-õpik.
5. Väga põhjalik lahendusmeetod ja selle kirjeldus ülesandele ‘rekursiivne otsing – alamsõne indeks sõnes’;
    1. “Rekursiooni kasutades tuleb meetodis määrata tingimusi, kui rekursiooni ei pea kasutama, muidu meetod hakkab kutsuma ise ennast välja lõpmatult palju ja see tekitab StackOverFlowError’i.”
6. Kasutati (täpselt) samu ülesandeid, kui eelnevas programmeerimise kursuses. Oleks võinud vähemalt harjutuse vormis olla.
7. Õpikust:
    1. Probleemilahendamine kasutades rekursiooni (õpik):
        1. “All recursive methods have the following characteristics:
            - The method is implemented using an if-else or a switch statement that leads to different cases.
            - One or more base cases (the simplest case) are used to stop recursion.
            - Every recursive call reduces the original problem, bringing it increasingly closer to a base case until it becomes that case.

            In general, to solve a problem using recursion, you break it into subproblems. Each subproblem is the same as the original problem but smaller in size. You can apply the same approach to each subproblem to solve it recursively.”
        2. Päriselulised näited: kohvijoomine (jood kuni tass on tühi), sõnumi printimine kuni on prinditud n korda.
        3. Rekursioon vs iteratsioon (õpik): “The decision whether to use recursion or iteration should be based on the nature of, and your understanding of, the problem you are trying to solve. The rule of thumb is to use whichever approach can best develop an intuitive solution that naturally mirrors the problem. If an iterative solution is obvious, use it. It will generally be more efficient than the recursive option.”
        4. Sabarekursiooni headuse kohta: “Tail recursion is desirable: because the method ends when the last recursive call ends, there is no need to store the intermediate calls in the stack. Compilers can optimize tail recursion to reduce stack size.”
        5. Hea: ülesanded on märgitud tärniga olenevalt nende raskusastmest lihtne – ilma tärnita, keskmine (*), raske (**) ja keerukas (***);
        6. Hea: samu ülesandeid rakendatakse erinevate andmetüüpidega (sõne, massiiv, täisarv);

7. Hea: iga ülesanne lõppeb käsuga “Write a test program that prompts the user to…”;

ITI0204 Algoritmid ja andmestruktuurid (3. semester)

- Õpiväljundid ja aine sisu 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:
    1. Rekursiooni ei mainita.
- Õppeviisid:
    1. loengud, praktikumid, harjutused
- Õppematerjalid:
    1. R. Neapolitan, K. Naimipour, Foundations of Algorithms 5th edition, 2014
    2. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, 2009
    3. J. Kiho, Algoritmid ja andmestruktuurid
        1. Mainib rekursiooni, aga ei tegele eraldi selle õpetamisega.
- Tähelepanekuid ja häid ideid
    1. Külalistel ei ole õppematerjalidele ligipääsu.

Tallinna Ülikool

- Rekursiooni ei mainitud ühegi Informaatika õppekava kohustuslikus õppeaines. Ei võta valimisse.

Maailmaülikoolid

Valim ülikoolidest ja ATI ERASMUS+ partneritest

Times Higher Education portaali 2019. aasta statistika: https://www.timeshighereducation.com/world-university-rankings/2019/subject-ranking/computer-science#!/page/0/length/25/sort_by/rank/sort_order/asc/cols/stats

Tartu Ülikooli positsioon maailmas: 251-300.

Legend:
VH – rohkelt materjali, mida oleks TÜ omaga hea võrrelda.
H/H1 – hea materjal/üks hea materjal
K – head õpperaamatud/ kirjandus, millele viidatakse, kui ülikooli slaide/ ülesandeid aga ei ole.

Mat.	Erasmus	Positsioon	Ülikool	Riik/ Osariik
VH	-	301-400	Rensselaeri Polütehniline Instituut (RPI)	New York
K	+	251-300	Bremeni Ülikool	Saksamaa
K	-	251-300	Johannes Kepleri Ülikool Linzis	Saksamaa
K	+	201-250	Konstanzi Ülikool	Saksamaa
VH	-	201-250	Bergeni Ülikool	Norra
K	-	201-250	Genti Ülikool	Belgia
K	+	176-200	Ulmi Ülikool	Saksamaa
K		151-175	Aalborgi Ülikool	Taani
VH	+	126-150	Kataloonia Polütehniline Ülikool (KPÜ)	Hispaania
H1	+	126-150	Helsinki Ülikool	Soome
VH	-	101-125	Chalmersi Ülikool	Rootsi
VH	+	88	Aalto Ülikool	Soome
VH	-	44	Marylandi Ülikool, College Park	Maryland
VH	-	11*	California Ülikool, Berkeley	California
VH	-	8	Princetoni Ülikool	New Jersey
VH	-	5	Massachusettsi Tehnoloogiainstituut (MIT)	Massachusetts

* paremusjärjestuses kategooriaga „*Engineering & technology*“

Rensselaeri Polütehniline Instituut (RPI)

Õppekava 2018/19 sisseastunutele:
https://rpi.app.box.com/s/nem1riuz0ln0fxf9dzf3k0ufbg1y0fys

CSCI 1100 - Computer Science I (1. semester)

Kursuse koduleht:
https://www.cs.rpi.edu/academics/courses/fall18/csci1100/lecture_notes.html

- Õpiväljundid ja aine sisu:
  - Rekursiooni otseselt ei mainita. Üldiselt programmeerimisest, mida saab mõista ka rekursiooni kontekstis.
    1. *“Demonstrate proficiency in the purpose and behavior of basic programming constructs.*
    2. *Design algorithms and programs to solve small-scale computational programs*
    3. *Write, test and debug small-scale programs*
    4. *Demonstrate an understanding of the wide-spread application of computational thinking to real-world problems.”*
- Õppeviisid:
  - Üks loeng, praktikumid (ingl *labs*)
- Õppematerjalid:

- Õpik: *Practical Programming: An Introduction to Computer Science Using Python* by Campbell, Gries, and Montojo (2nd edition!)
- 23. loeng e-õpikus: https://www.cs.rpi.edu/academics/courses/fall18/csci1100/lecture_notes/lec23_recursion.html
- 23. loengu harjutused e-õpikus: https://www.cs.rpi.edu/academics/courses/fall18/csci1100/lecture_notes/lec23_recursion_exercises/exercises.html
- Lahendused viimasele kolmele ülesandele (teisel lehel): https://www.cs.rpi.edu/academics/courses/fall18/csci1100/class_code.html#lecture-23

- Esitatud ülesanded
  - Kõik näited ja ülesanded:
    1. [N, H] täisarvu vähendamine;
    2. [N, H] järjendi elementide printimine;
    3. [N, H] Fibonacci (olemust kirjeldav näide);
    4. [H] rekursiivse funktsiooni (maksimaalne väärtus järjendis) täiendamine (lisada järjendi väärtused kokku);
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal
    1. rekursiivselt lahendatud probleemi iteratiivselt lahendamine:
       1. [H] Fibonacci;
    2. mõttes etteantud funktsiooni väärtustamine olenevalt sisendist:
       1. [N, H] täisarvu vähendamine;
       2. [N, H] järjendi elementide printimine;
    3. baasjuhu ja sammu äratundmine ning mõistmine:
       1. [N, H] täisarvu vähendamine;
    4. rekursiooni vähenemine baasjuhu suunas:
       1. [N, H] täisarvu vähendamine;
       2. [N, H] järjendi elementide printimine;
    5. koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine:
       1. [H] rekursiivse funktsiooni (maksimaalne väärtus järjendis) täiendamine (lisada järjendi väärtused kokku);
    6. algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
       1. [N, H] Fibonacci (olemust kirjeldava näite põhjal);
    7. iteratiivse ja rekursiivse koodi ajalise keerukuse vahe:
       1. [H] Fibonacci (suure n-i korral);
- Tähelepanekuid ja häid ideid
  - Rekursiooni käsitleti alles 23. loengus (kokku 24 loengut).
  - E-õppematerjali alguses on kokkuvõte sellest, millest rääkima hakatakse. Lõpus on veidi põhjalikum kokkuvõte põhilisest, mida meelde jätta,
  - Kohe alguses põhjalik selgitus, mis toimub rekursiivse funktsiooni väljakutsel Python'i *call stack* mehhanismis (23. loeng e-õpikus);
  - Faktoriaali kasutamine näitena on hea, sest faktoriaali definitsioon $n!=n(n-1) \cdot (n-2) \cdots 1$, kus $0! = 1$, on ebatäpne, sest "…" ei ole formaalselt defineeritud.

Rekursiivselt faktoriaali kirjutamine lahendab selle mure: n! = n·(n−1)!, kus 0! = 1. (23. loeng e-õpikus);

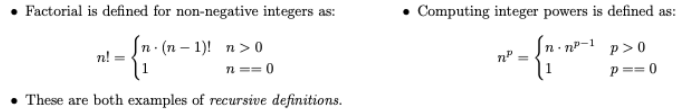
- "Guidelines for Writing Recursive Functions
    1. Define the problem you are trying to solve in terms of smaller / simpler instances of the problem. This includes
        1. What needs to happen before making a recursive call?
        2. What recursive call(s) must be made?
        3. What needs to happen to combine or generate results after the recursive call (or calls) ends?
    2. Define the base case or cases.
    3. Make sure the code is proceeding toward the base case in every step." (23. loeng e-õpikus);
- Rekursiooni ohud:
    1. "Some recursive function implementations contain wasteful repeated computation.
    2. Recursive function calls — like any function calls — typically involve hidden overhead costs.
    3. Often, therefore, a recursive function can (and should) be replaced with a non-recursive, *iterative* function that is significantly more efficient.
    4. This is easy to do for both Factorial and Fibonacci, as we will see in class." (23. loeng e-õpikus);
    5. "Looking carefully at the Fibonacci definition shows that calculating Fibonacci number f_{n-1} requires calculating Fibonacci number fn−2, which is also required for calculating Fibonacci number fn. This means there is redundant computation. This redundancy gets worse for numbers fn−3, fn−4, etc." (23. loengu harjutused e-õpikus);
- Põhjuseid, miks rekursiooni õppida ja kasutada (23. loeng e-õpikus):
    1. "Many of our definitions and even, our logical structures (such as lists), are formalized using recursion.
    2. Sometimes recursive functions are the first ones we come up with and the easiest to write (at least after you are comfortable with recursion).
    3. Only later do we write non-recursive versions.
    4. Sometimes on harder problems it is difficult to even write non-recursive functions! The list flattening problem below is one such example."
- *Merge sort*i fundamentaalne idee on rekursiivne. (23. loeng e-õpikus);
    1. Any list of length 1 is sorted
    2. Otherwise:
        1. Split the list in half
        2. Recursively sort each half
        3. Merge the resulting sorted halves

    "Comparing what we write to our earlier non-recursive version of merge sort shows that the primary job of the recursion is to organize the merging process!"

CSCI 1200 - Data Structures (2. semester)

Kursuse koduleht: https://www.cs.rpi.edu/academics/courses/spring17/ds/
Vol 2: https://www.cs.rpi.edu/academics/courses/fall18/csci1200/index.php
Vol 3: https://www.cs.rpi.edu/academics/courses/spring18/csci1200/index.php

- Õpiväljundid ja aine sisu https://www.cs.rpi.edu/academics/courses/spring17/ds/syllabus.php:
  - Aine eelduseks on "Algorithmic concepts:
    1. You should have seen and have a basic understanding of the following concepts:
       1. Recursion
       2. Run time analysis of algorithms (big O notation)
       3. Elementary searching and sorting algorithms"
  - Õpiväljundites rekursiooni otseselt ei mainita. Küll aga mainitakse järgnevat:
    1. "Analyze the performance of algorithms and data structures.
    2. Design and implement efficient customized data structures."
- Õppeviisid:
  - loengud, praktikum
- Õppematerjalid:
  - Kohustuslikke raamatuid ei ole.
  - 7. praktikum: https://www.cs.rpi.edu/academics/courses/fall18/csci1200/labs/07_recursion/lab_post.pdf
  - 8. loeng: https://www.cs.rpi.edu/academics/courses/fall18/csci1200/lectures/08_big_o_notation_recursion.pdf
  - 10. loeng: https://www.cs.rpi.edu/academics/courses/fall18/csci1200/lectures/10_linked_lists.pdf
  - 13. loeng: https://www.cs.rpi.edu/academics/courses/fall18/csci1200/lectures/13_problem_solving_I.pdf
- Esitatud ülesanded
  - Kõik ülesanded ja näited:
    1. [N] faktoriaali ja täisarvu astendamise rekursiivsed definitsioonid[1]:



[1] https://www.cs.rpi.edu/academics/courses/fall18/csci1200/lectures/08_big_o_notation_recursion.pdf

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

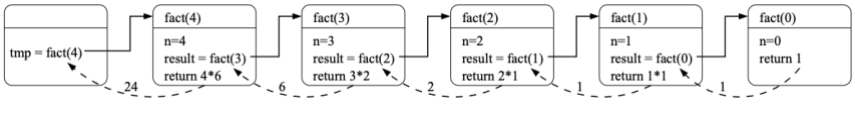
```
int intpow(int n, int p) {
  if (p == 0) {
    return 1;
  } else {
    return n * intpow( n, p-1 );
  }
}
```

2. [H] aktiveerimiskirjete joonistamine funktsiooni väljakutse põhjal (cout << intpow(4, 4) << endl, kus intpow() astendab rekursiivselt täisarvu);
3. [H] intpow() rekursiivselt lahendatud probleemi iteratiivselt lahendamine;
4. [H] intpow() rekursiivse ja iteratiivse funktsiooni ajalise keerukuse leidmine (O-relatsiooni kasutades);
5. [N, H] vektori sisu printimine;
6. [H] vektori sisu tagurpidi printimine eelneva näite ja harjutuse põhjal;
7. [H] Fibonacci (rekursiivne ja iteratiivne versioon);
8. [N, H] binaarotsing, et leida etteantud vektorist väärtust "x" (käsitletakse nii 8. kui ka korratakse uuesti üle 12. loengus);
9. [H] mittelineaarne sõnaotsing vektoritest koosnevast maatriksist (12. loeng, kordamine 13. loengus);
10. [H] kõik perfektsed numbrid (ingl *perfect number*) alla sisendväärtuse n-i;
11. [H] kahe lähima ujukomaarvu leidmine n-suuruselisest ujukomaarvude jadast;
12. [H] numbrijadast korduste eemaldamine;
13. [H] x ja y-telgedega ruudujoonestikus kõikvõimalike lühenevate teekondade leidmine suvalisest punktist (x,y) algpunkti (0,0) (7. praktikum);

- Ülesannete klassifikatsioon tehniliste oskuste põhjal
    1. rekursiivselt lahendatud probleemi iteratiivselt lahendamine:
        1. [H] intpow() rekursiivselt lahendatud probleemi iteratiivselt lahendamine (x2);
        2. [N, H] Binaarotsing, et leida etteantud vektorist väärtust "x";
    2. mõttes etteantud funktsiooni väärtustamine olenevalt sisendist:
        1. [H] vektori sisu printimine;
    3. baasjuhu ja sammu äratundmine ning mõistmine:
        1. [N, H] Binaarotsing, et leida etteantud vektorist väärtust "x";
        2. [H] x ja y-telgedega ruudujoonestikus kõikvõimalike lühenevate teekondade leidmine suvalisest punktist (x, y) algpunkti (0,0);
    4. alamülesandest terve ülesande lahenduseni jõudmine:
        1. [H] x ja y-telgedega ruudujoonestikus kõikvõimalike lühenevate teekondade leidmine suvalisest punktist (x, y) algpunkti (0,0);
    5. rekursiooni vähenemine baasjuhu suunas:
        1. [H] x ja y-telgedega ruudujoonestikus kõikvõimalike lühenevate teekondade leidmine suvalisest punktist (x, y) algpunkti (0,0);
    6. funktsiooni käitumine ootamatute argumentide korral:
        1. [H] x ja y-telgedega ruudujoonestikus kõikvõimalike lühenevate teekondade leidmine suvalisest punktist (x, y) algpunkti (0,0);

7. koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine:
    1. [H] vektori sisu tagurpidi printimine eelneva näite ja harjutuse põhjal;
    2. [N, H] binaarotsing, et leida etteantud vektorist väärtust “x” (if-else struktuuri rekursiivses funktsioonis teise if-else struktuuriga asendamine) (x2);
    3. [H] mittelineaarne sõnaotsing vektoritest koosnevast maatriksist;
8. algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
    1. [H] Fibonacci;
9. rekursiooni ajaline keerukus:
    1. [H] mittelineaarne sõnaotsing vektoritest koosnevast maatriksist (O-relatsiooni kasutades);
10. iteratiivse ja rekursiivse koodi ajalise keerukuse vahe:
    1. [H] intpow() rekursiivse ja iteratiivse funktsiooni ajalise keerukuse leidmine (O-relatsiooni kasutades) (x2).
    2. [H] Fibonacci;
11. iteratiivse ja rekursiivse koodi mälukasutuse vahe:
    1. [H] Fibonacci;
12. aktiveerimiskirjete ja/või rekursioonipuu joonistamine funktsiooni väljakutse põhjal:
    1. [H] aktiveerimiskirjete joonistamine funktsiooni väljakutse põhjal (cout << intpow(4, 4) << endl);
13. rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine:
    1. [H] kõik perfektsed numbrid (ingl *perfect number*) alla sisendväärtuse n-i;
    2. [H] kahe lähima ujukomaarvu leidmine n-suuruselisest ujukomaarvude jadast;
    3. [H] numbrijadast korduste eemaldamine;

- Tähelepanekuid ja häid ideid
    - Kursust õpetatakse C++ keelega.
    - 8. loengu plaan on hea:
        1. Algorithm Analysis / Computational Complexity
        2. Orders of Growth, Formal Definition of Big O Notation
        3. Simple Recursion
        4. Visualization of Recursion
        5. Iteration vs. Recursion
        6. “Rules” for Writing Recursive Functions
        7. Lots of Examples!
    - Illustratsioon, kuidas rekursiivse funktsiooni väljakutsemehhanism töötab (*activation record*) (8. loeng):



Eraldi välja toodud "We've already been drawing pictures of the stack and the local variables that functions place on the stack. Once we leave a function (or any scope delimited with { ... }), the local variables on the stack are cleaned up and we cannot / should not try to access them again."

- Iteratsioon vs rekursioon (8. loeng):
  - Faktoriaali funktsioon iteratiivselt.
  - "Often writing recursive functions is more natural than writing iterative functions, especially for a first draft of a problem implementation.
  - You should learn how to recognize whether an implementation is recursive or iterative, and practice rewriting one version as the other. Note: We'll see that not all recursive functions can be easily rewritten in iterative form!
  - Note: Big O notation for the number of operations for the recursive and iterative versions of an algorithm is usually the same. However in C, C++, Java, and some other languages, iterative functions are generally faster than their corresponding recursive functions. This is due to the overhead of the function call mechanism. Compiler optimizations will sometimes (but not always!) reduce the performance hit by automatically eliminating the recursive function calls. This is called tail call optimization."
  - Kõike eelnevat korratakse üle 12. loengus.
- Reeglid rekursiivsete funktsioonide kirjutamiseks (8. loeng):
  - "Here is an outline of five steps that are useful in writing and debugging recursive functions. Note: You don't have to do them in exactly this order...
    1. Handle the base case(s).
    2. Define the problem solution in terms of smaller instances of the problem. Use wishful thinking, i.e., if someone else solves the problem of fact(4) I can extend that solution to solve fact(5). This defines the necessary recursive calls. It is also the hardest part!
    3. Figure out what work needs to be done before making the recursive call(s).
    4. Figure out what work needs to be done after the recursive call(s) complete(s) to finish the computation. (What are you going to do with the result of the recursive call?)
    5. Assume the recursive calls work correctly, but make sure they are progressing toward the base case(s)!"
    6. Kõik eelnev korratakse üle ka 12. loengus!
- "It is common to have a driver function that just initializes the first recursive function call."
- Lihtahela definitsioon. "The definition is recursive: A linked list is either: – Empty, or – Contains a node storing a value and a pointer to a linked list." (10. loeng)
- Binaarotsinguga alustatakse 8. loengus. Kolm ülesannet korratakse 12. loengus uuesti üle.

- 13. loeng on pühendatud probleemilahendusoskuste õppimisele. Välja tuuakse kolm sammu/ faasi, kuidas algoritme disainida ja implementeerida (13. loeng):
  - "Generating and Evaluating Ideas
    - Most importantly, play with examples! Can you develop a strategy for solving the problem? You should try any strategy on several examples. Is it possible to map this strategy into an algorithm and then code?
    - Try solving a simpler version of the problem first and either learn from the exercise or generalize the result.
    - Does this problem look like another problem you know how to solve?
    - If someone gave you a partial solution, could you extend this to a complete solution?
    - What if you split the problem in half and solved each half (recursively) separately?
    - Does sorting the data help?
    - Can you split the problem into different cases, and handle each case separately?
    - Can you discover something fundamental about the problem that makes it easier to solve or makes you able to solve it more efficiently?
    - Once you have an idea that you think will work, you should evaluate it: will it indeed work? are there other ways to approach it that might be better / faster? if it doesn't work, why not?
  - Mapping Ideas into Code
    - How are you going to represent the data? What is most efficient and what is easiest?
    - Can you use classes to organize the data? What data should be stored and manipulated as a unit? What information needs to be stored for each object? What operations (beyond simple accessors) might be helpful?
    - How can you divide the problem into units of logic that will become functions? Can you reuse any code you're previously written? Will any of the logic you write now be re-usable?
    - Are you going to use recursion or iteration? What information do you need to maintain during the loops or recursive calls and how is it being "carried along"?
    - How effective is your solution? Is your solution general? How is the performance? (What is the order notation of the number of operations)? Can you now think of better ideas or approaches?
    - Make notes for yourself about the logic of your code as you write it. These will become your invariants; that is, what should be true at the beginning and end of each iteration / recursive call.
  - Getting the Details Right

        - Is everything being initialized correctly, including boolean flag variables, accumulation variables, max / min variables?
        - Is the logic of your conditionals correct? Check several times and test examples by hand.
        - Do you have the bounds on the loops correct? Should you end at n, n − 1 or n − 2?
        - Tidy up your "notes" to formalize the invariants. Study the code to make sure that your code does in fact have it right. When possible use assertions to test your invariants. (Remember, sometimes checking the invariant is impossible or too costly to be practical.)
        - Does it work on the corner cases; e.g., when the answer is on the start or end of the data, when there are repeated values in the data, or when the data set is very small or very large?"
    - (13. loeng) Loengu lõpus on veel häid probleemilahendamise strateegiaid.
    - 7. praktikumi on jaotatud ülesannete põhjal kontrollpunktideks. Iga kontrollpunktile on määratud umbkaudne aeg, kui palju ülesanne aega võiks võtta.

CSCI 2200 - Foundations of Computer Science (3. semester)

Kursuse koduleht: https://www.cs.rpi.edu/academics/courses/spring17/focs/

- Õpiväljundid ja aine sisu:
  https://www.cs.rpi.edu/academics/courses/spring17/focs/general-info.pdf
    - "Upon successful completion of this course, each student is able to: … apply mathematical tools such as induction and recursion"
- Õppeviisid:
    - Loengud (16. ja 17. loeng)
      https://www.cs.rpi.edu/academics/courses/spring17/focs/slides/March21.pdf
      https://www.cs.rpi.edu/academics/courses/spring17/focs/slides/March24.pdf
- Õppematerjalid:
    - Kenneth H. Rosen, Discrete Mathematics and Its Applications, 7th ed., McGraw Hill, 2012
- Esitatud ülesanded
    - Kõik ülesanded:
      ~~1. [N] faktoriaal;~~
      ~~2. [N] arvu a astendamine arvuga n;~~
      ~~3. [N] Fibonacci;~~
    - Ülesannete klassifikatsioon tehniliste oskuste põhjal
        1. baasjuhu ja sammu äratundmine ning mõistmine:
            1. [N] faktoriaal;
            2. [N] arvu a astendamine arvuga n;
        2. rekursiooni vähenemine baasjuhu suunas:
            1. [N] faktoriaal;
            2. [N] arvu a astendamine arvuga n;
        3. iteratiivse ja rekursiivse koodi ajalise keerukuse vahe:
            1. [N] Fibonacci;

4. rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine:
    1. [N] Fibonacci;

- Tähelepanekuid ja häid ideid:
    - Slaididel on näidatud miks iteratiivne Fibonacci on efektiivsem kui rekursiivne.
    - Keskendub rekursioonile matemaatilisest perspektiivist. Küll aga korrati 17. loengu alguses rekursiivse algoritmi definitsioon koos mõne näitega üle.

CSCI 2300 - Introduction to Algorithms (4. semester)

Kursuse koduleht: https://www.cs.rpi.edu/~eanshel/CSCI2300.html
Vol2: https://www.cs.rpi.edu//~moorthy/Courses/CSCI2300/

- Õpiväljundid ja aine sisu https://www.cs.rpi.edu/~eanshel/2300/2300syllabus.html:
    - Rekursiooni ei tooda eraldi välja.
    - Eeldatakse, et tudeng, kes ainet võtab on juba kompetentne programmeerija, sest tegu ei ole programmeerimisainega. Loengutes ei arutleta väidetavalt koodi üle.
- Õppeviisid:
    - Praktikumid (3. praktikum) https://www.cs.rpi.edu//~moorthy/Courses/CSCI2300/lab3-2015.html
- Õppematerjalid:
    - *Algorithms*, Dasgupta, Papadamitriou and Vazirani McGraw Hill, 2008
    - *Introduction to Algorithms*, Cormen, Leiserson, Rivest and Stein Most Recent Edition McGraw Hill
    - Loengumaterjalidele ligipääsu ei ole.
- Esitatud ülesanded
    - Kõik ülesanded:
        - [H] mitme järjestikuse väljakutsega rekursiooni töö ennustamine (x2);
    - Ülesannete klassifikatsioon tehniliste oskuste põhjal
        - mõttes etteantud funktsiooni väärtustamine olenevalt sisendist:
        - [H] mitme järjestikuse väljakutsega rekursiooni töö ennustamine (x2);
- Tähelepanekuid ja häid ideid
    - Raamat ei korda rekursiooni definitsiooni ega olemust.
    - Raamat võrdleb rekursiivse lahendusmeetodi keerukust teiste lahendusmeetoditega. Nt. lk. 12 põhjendab põhjalikult miks Fibonacci ei ole rekursiivselt hea mõte.

CSCI 4430 - Programming Languages (6. või 7. semester)

Kursuse koduleht: https://www.cs.rpi.edu/~milanova/csci4430/

- Õpiväljundid ja aine sisu:
    - Rekursiooni ei tooda eraldi välja.
- Õppeviisid:
    - loengud, kodutööd, kohustuslik lugemine

- Õppematerjalid:
    - Programming Language Pragmatics, 4th Edition, by Michael Scott, Morgan Kaufmann, 2015
    - 12. loeng: https://www.cs.rpi.edu/~milanova/csci4430/Lecture12.pdf
    - 13. loeng: https://www.cs.rpi.edu/~milanova/csci4430/Lecture13.pdf
    - Kõik loengud: https://www.cs.rpi.edu/~milanova/csci4430/schedule.htm
- Esitatud ülesanded
    - Kõik ülesanded:
        - [N, H] järjendi pikkus;
        - [N, H] järjendi teisse järjendisse lisamine;
        - [H] järjendis olevate järjendite arvu loendamine;
        - [H] järjendis olevate *atom*'ite (Scheme) kokku lugemine;
        - [H] järjendi printimine Scheme'i car ja cdr kasutades;
        - [H] n-tasemelise järjendi ühetasemeliseks muutmine;
    - Ülesannete klassifikatsioon tehniliste oskuste põhjal:
        - baasjuhu ja sammu äratundmine ning mõistmine
            - [H] järjendis olevate *atom*'ite (Scheme) kokku lugemine;
        - koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine:
            - [N, H] järjendi pikkus;
        - algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
            - [H] järjendis olevate järjendite arvu loendamine;
        - fikseerimata tasemete arvuga andmestruktuuride töötlemine:
            - [H] n-tasemelise järjendi ühetasemeliseks muutmine;
        - madal rekursioon:
            - [N, H] järjendi pikkus;
            - [N, H] järjendi teisse järjendisse lisamine;
            - [H] järjendis olevate *atom*'ite (Scheme) kokku lugemine;
        - sügav rekursioon:
            - [H] järjendi printimine Scheme'i car ja cdr kasutades;
- Tähelepanekuid ja häid ideid
    - Rekursiooni definitsiooni ei korrata.
    - Toodi sisse madala ja sügava rekursiooni mõiste.

Bremeni Ülikool

Moodulite raamat: https://www.szi.uni-bremen.de/wp-content/uploads/2018/05/mhb_InfB_2018.pdf

BA-600.01 Mathematische Grundlagen 1: Logik und Algebra (Mathematics 1) (1. semester)

- Õpiväljundid ja aine sisu (moodulite raamat):
    - Sisu (saksa kl Inhalte):

- (2) Denken: ... Widerspruch, Kontraposition, Rekursion
- (6) Lösen: ... Rekursionen (Formale Potenzreihen)

- Õppematerjalid (moodulite raamat):
  - G. und S. Teschl, Mathematik für Informatiker - Band 1: Diskrete Mathematik und Lineare Agebra. Springer 2006.
  - P. Hartmann, Mathematik für Informatiker: ein praxisbezogenes Lehrbuch. Vieweg+Teubner, 5. Auflage 2012.
  - E. Lehmann, F. Thomson Leighton, A.R. Meyer, Mathematics for computer science. MIT Skript 2011, Creative Commons (kostenlos online).
  - W.Doerfler,W.Peschek: Einführung in die Mathematik für Informatiker. Hanser Verlag 1988
  - Ch.Meinel,M.Mundhenk: Mathematische Grundlagen der Informatik, 2.Auflage, Teubner Verlag 2002.
  - R.L.Graham,D.E.Knuth,O.Patashnik: Concrete Mathematics. A Foundation for Computer Science.Addison-Wesley Publ.Co.1988

BA-700.01 Praktische Informatik 1: Imperative Programmierung und Objektorientierung (Practical Computer Science 1) (1. semester)

- Õpiväljundid ja aine sisu (moodulite raamat):
  - Inhalte:
    - "1. Datenstrukturen: ... rekursive Datentypen…
    - 3. Algorithmen: … Rekursion
  - Grundkomponenten imperativer Programmiersprachen: … Rekursion"
- Õppeviisid:
  - 5. praktikum http://www.informatik.uni-bremen.de/agbs/lehre/ws0809/pi1/aufgaben/serie5.pdf
- Õppematerjalid (moodulite raamat):
  - Saake und K.-U. Sattler: Algorithmen und Datenstrukturen. dpunkt.verlag, Heidelberg (2004)
  - R. Schiedermeier: Programmieren mit Java. Pearson, München (2005)
  - Veebimaterjalid (täpsem sessioonikava, exercise sheets, loenguslaidid puuduvad): http://www.informatik.uni-bremen.de/agbs/lehre/ws0809/pi1/
- Esitatud ülesanded:
  - Kõik ülesanded:
    - [H] järjestamise kiirmeetodi (ahelal) implementeerimine;
    - [H] küülikute ülesanne (mitu küülikut on 12 kuu pärast?);
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal:
    - rekursiivselt lahendatud probleemi iteratiivselt lahendamine:
      - [H] küülikute ülesanne (mitu küülikut on 12 kuu pärast?) (rekursiivne ja iteratiivne lahendus);
    - mõttes etteantud funktsiooni väärtustamine olenevalt sisendist:
      - [H] järjestamise kiirmeetodi (ahelal) implementeerimine;
    - rekursiooni ajaline keerukus:

- [H] järjestamise kiirmeetodi (ahelal) implementeerimine;

BA-700.03 Praktische Informatik 3: Funktionale Programmierung (Practical Computer Science 3) (3. semester)

- Õpiväljundid ja aine sisu (moodulite raamat):
  - "Inhalte: 1. Grundlagen der funktionalen Programmierung: Rekursion – Definition von Funktionen durch rekursive Gleichungen und Mustervergleich (pattern matching)"
- Õppeviisid:
  - Loengud, praktikumid (1., 2., 7. praktikum), praktikumid grupitööna
- Õppematerjalid (moodulite raamat):
  - Simon Thompson: Haskell - The Craft of Functional Programming, Addison-Wesley, 3. Auflage 2011.
  - Peter Pepper: Funktionale Programmierung. Springer-Verlag 1999.
  - Veebimaterjalid (slaidid, raamatud, exercise sheets): http://www.informatik.uni-bremen.de/~clueth/lehre/pi3.ws18/lectures.html
  - 8. praktikum: http://www.informatik.uni-bremen.de/~clueth/lehre/pi3.ws18/ueb/uebung-08.pdf
  - Kõik slaidid: http://www.informatik.uni-bremen.de/~clueth/lehre/pi3.ws18/lectures/handouts-all.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-07.pdf http://www.informatik.uni-bremen.de/~clueth/lehre/pi3.ws18/ueb/uebung-10.pdf
- Esitatud ülesanded
  - Kõik ülesanded:
    1. [H] kõikide arvude summa ühest sisendparameeter n-ini (1. praktikum);
    2. [H] ristsumma (nt. 457 -> 16) (2. praktikum);
    3. [H] kas String s on String t eesliide? (2. praktikum);
    4. [H] teksti joondamine (vasak, parem, mõlemast äärest) (7. praktikum);
    5. [H] suunamata graafi põhjal labürintide loomine (10. praktikum);
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal:
    1. baasjuhu ja sammu äratundmine ning mõistmine:
       1. [H] kõikide arvude summa ühest sisendparameeter n-ini (1. praktikum);
       2. [H] ristsumma (nt. 457 -> 16);
       3. [H] kas String s on String t eesliide?;
       4. [H] teksti joondamine (vasak, parem, mõlemast äärest);
       5. [H] suunamata graafi põhjal labürintide loomine;
    2. alamülesandest terve ülesande lahenduseni jõudmine:
       1. [H] teksti joondamine (vasak, parem, mõlemast äärest);
       2. [H] suunamata graafi põhjal labürintide loomine;
    3. rekursiooni vähenemine baasjuhu suunas:
       1. [H] teksti joondamine (vasak, parem, mõlemast äärest);
- Tähelepanekuid ja häid ideid

- NB! Loenguslaididel veel igasuguseid näiteid. Kõiki ei ole nende lihtsuse tõttu kaasatud hetkel.
- Sarnane UT Programmeerimiskeeled kursusega.
- Juba 1. semestril käsitletakse rekursiivseid andmetüüpe!
- Grupitööna vasakule kaldu kuhja (*leftist heap*) ja Huffmani puu käsitlemine. Rekursiooniga seoses suuri õpetussõnu ei anta (8. praktikum).
- Sõnum: sabarekursioon on mõistlik (5. loeng);
- Rekursiooni vormid: foldr = strukturaalne rekursioon, foldl = iteratsioon.

BA-601.02 Theoretische Informatik 2: Berechenbarkeitsmodelle und Komplexität (Theoretical Computer Science 2) (4. semester)

- Õpiväljundid ja aine sisu (moodulite raamat):
  - Inhalte: 1) Berechenbarkeit
    - "Primitiv rekursive Funktionen und -rekursive Funktionen"
    - "Semi-Entscheidbarkeit und Rekursive Aufzahlbarkeit"
- Õppematerjalid (moodulite raamat):
  - J.E. Hopcroft, R. Motwani, J.D. Ullman: Einfuhrung in die Automatentheorie, Formale Sprachen und Komplexitatstheorie, Pearson Studium 2011
  - J.E. Hopcroft, R. Motwani, J.D. Ullman: Introduction to Automata Theory, Languages, and Computation (3rd edition). Pearson Education, 2014
  - C. Lutz: Theoretische Informatik 2, Skript

Johannes Kepleri Ülikool Linzis

Õppekava: http://cs.jku.at/teaching/bachelor/
Moodle kursuseotsing (hetkel midagi kasulikku leidnud ei ole):
https://moodle.jku.at/jku2015/course/search.php?search=algorit

[INBIPUEALG1] UE Algorithms and Data Structures 1 (2. semester)

- Õpiväljundid ja aine sisu:
  https://studienhandbuch.jku.at/detail.php?lang=de&klaId=INBIPUEALG1 ja
  http://ssw.jku.at/Teaching/Lectures/Algo/
  - Õpiväljundites rekursiooni ei mainita.
  - Aine sisu: "Recursion:
    Term, classification, scheduling, applications, conversion of recursive algorithms in iterative and vice versa"
- Õppematerjalid:
  - Põnev materjal: andmestruktuuride visualiseerimine (University of San Francisco)
    https://www.cs.usfca.edu/~galles/visualization/Algorithms.html
  - Loenguslaidid (slaide rekursioonist veel ei ole):
    https://www.pervasive.jku.at/Teaching/lvaInfo.php?key=602&do=material

- Kursuse leht (natuke täpsemalt sisust, aga materjalideta): http://ssw.jku.at/Teaching/Lectures/Algo/

[INBIPVOBEKO] VL predictability and complexity - Berechenbarkeit und Komplexität (3. semester)

- Õpiväljundid ja aine sisu https://studienhandbuch.jku.at/detail.php?lang=de&klaId=INBIPVOBEKO:
  - Õpiväljundid: "Komplexitätsanalyse von iterativen und rekursiven Algorithmen."
  - Sisu: "primitiv und mu-rekursive Funktionen"
- Õppematerjalid:
  - Wolfgang Schreiner: "Computability and Complexity" (Skript).
  - Dirk W. Hoffmann: Theoretische Informatik.
  - John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman: Introduction to Automata Theory, Languages and Computation (deutsche Ausgabe: Einführung in Automatentheorie, Formale Sprachen und Berechenbarkeit).
  - Michael Sipser: Introduction to the Theory of Computation.
  - Thomas H. Cormen et al: Introduction to Algorithms.

Konstanzi Ülikool

Õppekava: https://www.uni-konstanz.de/uploads/tx_studiengang/download_71_1521193449.pdf
Informaatika õppekava moodulite raamat: https://www.informatik.uni-konstanz.de/uploads/tx_studiengang/en_modulebook_178_1508485273.pdf

5.2.1 Konzepte der Informatik / Principles of Computer Science 1 (1. semester)

- Õpiväljundid ja aine sisu (moodulite raamat):
  - "Eigenschaften von Algorihmen, insbesondere Algorithmenkomplexität und Korrektheit, sowie die algorithmische Konzepte Iteration und Rekursion"
- Õppematerjalid (moodulite raamat):
  - Herold, Lurz & Wohlrab: Grundlagen der Informatik (lbs 830/h27, ISBN 3-8273-7305-2)
  - Grumm & Sommer: Einführung in die Informatik (lbs 830/g95(7), ISBN 3-486-58115-7)
  - Küchlin & Weber: Einführung in die Informatik (lbs 843/k92(3), ISBN 3-540-20958-1)
  - Cormen: Algorithmen - eine Einführung (lbs 840/a53, ISBN 3-486-58262-8) oder das englische Original Introduction to Algorithms (lbs 830/c67(28), ISBN 0-262-03293-7)
  - Sedgewick: Algorithmen in Java (Teil 1-4) (kid 112:n/s26-1/4, ISBN 3-8273-7072- 8) oder das englische Original inklusive Graphalgorithmen: Algorithms in

Java, Part 1-4 (ISBN 0-201-36120-9), Algorithms in Java, Part 5 (ISBN 0-201-36121-6)
    - Ottmann & Widmayer: Algorithmen und Datenstrukturen (lbs 830/o99(4), ISBN 3-8274-1029-0)

5.4.4 Algorithmen und Datenstrukturen / Algorithms and Data Structures (2. semester)

- Õpiväljundid ja aine sisu (moodulite raamat):
    - Sisu: "Inhalte: In der Vorlesung werden Standardalgorithmen und grundlegende Datenstrukturen behandelt. …, recursive Algorithmen."
- Õppematerjalid (moodulite raamat):
    - N. Blum: Algorithmen und Datenstrukturen. Oldenbourg, 2004
    - T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein: Algorithmen - Eine Einführung. Oldenbourg, 2007 (2. Aufl.)
    - T. Ottmann, P. Widmayer: Algorithmen und Datenstrukturen. Spektrum Akademischer Verlag, 2002 (4. Aufl.)
    - P. Sanders, K. Mehlhorn: Algorithms and Data Structures. Springer, 2008
    - R. Sedgewick, K. Wayne: Algorithmen -Algorithmen und Datenstrukturen. Pearson, 2014 (4. Aufl.)
    - U. Schöning: Algorithmik. Spektrum Akademischer Verlag, 2001
    - M.A. Weiss: Data Structures and Algorithm Analysis in Java. Pearson, 2007 (2nd ed.)

5.5.2 Theoretische Grundlagen der Informatik / Theoretical Computer Science (4. semester)

- Õpiväljundid ja aine sisu (moodulite raamat):
    - Õpiväljundid: "Entscheidbarkeit und Berechenbarkeit: … μ-rekursive Funktionen, …"
- Õppematerjalid (moodulite raamat):
    - U. Schöning, Theoretische Informatik - kurzgefasst, Spektrum Akademischer Verlag, 4. Auflage, 2001.
    - J. E. Hopcroft, R. Motwani, J. D. Ullman, Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie, Pearson Studium, 2. Auflage, 2002.
    - Wegener, Theoretische Informatik - eine algorithmische Einführung, B. G. Teubner Verlag, 2. Auflage, 1999.

Bergeni Ülikool

- Õppekava: https://www.uib.no/en/studies/BAMN-DVIT/plan
- Kõikide kursuselehte otsing: https://mitt.uib.no/search/all_courses?search=inf102

INF 100 Introduction to programming (1. semester)

- Ei käsitle rekursiooni.

INF 101 Object-oriented programming 2 (2. semester)

- Õpiväljundid ja aine sisu:
    - 2017. versioonis: Sisu: "Rekursjon og rekursive datastrukturer
        1. Rekursjon og induksjon
        2. Rekursjonsbrønner
        3. Rekursiv listestruktur og listehåndtering
        4. Rekursiv sortering (merge sort)"
- Õppeviisid:
    - 6. praktikum https://retting.ii.uib.no/inf101.v17.oppgaver/inf101.v17.lab6/tree/master/src/inf101/v17/lab6/recursion
    - 14. loeng https://mitt.uib.no/courses/16313/files/folder/lectures?preview=1600287
    - 15. loeng https://mitt.uib.no/courses/16313/files/folder/lectures?preview=1602007
- Õppematerjalid:
    - Ametlikku raamatut ei ole.
    - Kursuse wiki: https://retting.ii.uib.no/inf101/inf101.v190/wikis/home
    - Kõik loengumaterjalid: https://mitt.uib.no/courses/16313/files/folder/lectures?
    - Java All-in-One For Dummies 4th Edition, Doug Lowe, peatükk 4: Using Recursion
    - Big Java, Late Objects, Cay Horstmann, peatükk 13: Recursion
    - Princetoni materjalid: https://introcs.cs.princeton.edu/java/23recursion/
    - Algoritmitöid illustreerivad veebilehed: http://sorting.at/, https://imgur.com/gallery/GD5gi
- Esitatud ülesanded
    - Kõik ülesanded:
        1. [HV] ristsumma (nt. 457 -> 16);
        2. [H] püramiidi kõrgusele vastav plokkide arv;
        3. [HV] kõikvõimalike kombinatsioonide arv: nCr(n,r), kus n on elementide arv ja r on iga kombinatsiooni suurus;
        4. [HL] sisendparameetri n ruutu arvutamine (selle põhjal, et saame võtta n-1 ja selle kahega korrutada);
        5. [HL] mõttes etteantud salapärase (mistery()) funktsiooni väärtustamine olenevalt sisendist;
        6. [H] malelauale 8 kuninganna asetamine nii, et nad üksteisele tuld ei annaks;
    - Ülesannete klassifikatsioon tehniliste oskuste põhjal
        - mõttes etteantud funktsiooni väärtustamine olenevalt sisendist:
            1. [HV] kõikvõimalike kombinatsioonide arv: nCr(n,r), kus n on elementide arv ja r on iga kombinatsiooni suurus;
            2. [HL] mõttes etteantud salapärase (mistery()) funktsiooni väärtustamine olenevalt sisendist (x2);

        - baasjuhu ja sammu äratundmine ning mõistmine:
            1. [HV] ristsumma;
            2. [H] püramiidi kõrgusele vastav plokkide arv;
            3. [H] sisendparameetri n ruutu arvutamine (selle põhjal, et saame võtta n-1 ja selle kahega korrutada);
            4. [H] malelauale 8 kuninganna asetamine nii, et nad üksteisele tuld ei annaks;
        - rekursiooni vähenemine baasjuhu suunas:
            1. [HV] ristsumma;
            2. [H] püramiidi kõrgusele vastav plokkide arv;
            3. [H] sisendparameetri n ruutu arvutamine (selle põhjal, et saame võtta n-1 ja selle kahega korrutada);
        - funktsiooni käitumine ootamatute argumentide korral:
            1. [H] püramiidi kõrgusele vastav plokkide arv (negatiivne kõrgus on ülesandekirjelduses eraldi defineeritud);
        - algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
            1. [HV] kõikvõimalike kombinatsioonide arv: nCr(n,r), kus n on elementide arv ja r on iga kombinatsiooni suurus;
            2. [H] malelauale 8 kuninganna asetamine nii, et nad üksteisele tuld ei annaks;
- Tähelepanekuid ja häid ideid
    - 6. praktikum mainis, et viimase ülesande nCr(n,r) lahendus võib suurte sisendite korral aeglane olla. Küsitakse, "Is there something we can do to speed it up?". Seega, otseselt keerukusest praktikumimaterjalides ei räägita.
    - Iga loengu alguses korratakse kiirelt üle, millega eelmine kord tegeleti.
    - Droste efekt rekursiivsest pildist (14. loeng): https://www.webdesignerdepot.com/2009/09/50-stunning-examples-of-the-droste-effect/.
    - "How to do recursion
        1. Identifying the simpler problem can be difficult -> this requires practice
        2. Afterwards, the simplest problem needs to be identified (base case)
        3. According to 'Big Java, Late Objects': 'key to the successful design of a recursive method is not to think about it" (14. loeng);
    - Kihvt tükk huumorit (14. loeng): "Microsoft interview question
        1. How would you explain recursion to a 6-year-old?
        2. Explain recursion to someone one year younger than you. Have them explain recursion to someone one year younger than them. Continue until you have a 7-year-old explaining recursion to a 6-year-old. Done."
    - Fibonacci (väljakutse puul (ingl *call tree*) korduste st. ebaefektiivsuse illustreerimine (14. loeng);

- Efektiivsusmure parandab dünaamiline programeerimine. (15. loeng): "Recursively divide a complex problem into a number of simpler subprograms and store the answer to the solution of each subproblem"
- Väidavad, et rekursioon on elegantsem (14. loeng).
- "When to use recursion
    1. Many problems: recursive solutions are easier to understand and implement correctly than their iterative counterparts.
        1. Sometimes there is no obvious iterative solution at all!
        2. If you are facing a problem where you can't determine how many nested for-loops you will need
        3. Also can draw cool fractals with it" (14. loeng);
- Tagasipöördumistehnika (ingl *backtracking*) (15. loeng):
    1. "Problem solving techniques that builds up partial solutions that get increasingly closer to the goal
    2. If cannot be completed: abandon and return to other candidates"
    3. Omadused:
        1. Procedure to examine a partial solution, and:
            1. – Accept is as a solution
            2. – Abandon it if it is 'illegal' or leading nowhere
            3. – Continue extending it
        2. Procedure to extend a partial solution, generating more solutions that come closer"
    4. Ülevalt-alla ja alt-üles dünaamilise programmeerimise tutvustamine
        1. 15.l "Store or cache the result of each subproblem that we solve, so that the next time we need it, we can use the cached values instead of solving the subproblem from scratch"
        2. "Compute solutions to all of the subproblems, starting with the "simplest" subproblems and gradually building up solutions to more and more complicated subproblems"

INF 122 Functional Programming (3. semester)

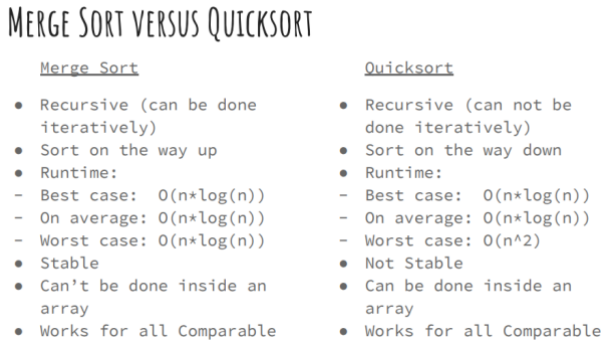
- Õpiväljundid ja aine sisu:
    - Õpiväljundid: Knowledge: "The student knows:"
        1. the basic concepts of functional programming, such as:
            - recursion
- Materjalile ei ole ligipääsu.

INF 102 Algorithms, Data Structures and Programming (3. semester)

- Õpiväljundid ja aine sisu:
    - Rekursiooni ei mainita õpiväljundites ega sisus.
- Õppeviisid:
    - Loengud, Youtube videotele viitamine
- Õppematerjalid:

- „Reading list“: https://mitt.uib.no/courses/12780/external_tools/118
- Robert Sedgewick and Kevin Wayne, Algorithms (4th ed), Addison-Wesley, 2011. Koduleheküg: http://algs4.cs.princeton.edu/home/
- Github: https://github.com/torsteins/inf102f18-lectures
- Kursuseleht: https://mitt.uib.no/courses/12780
- Loenguslaidid: https://mitt.uib.no/courses/12780/files/folder/lecturenotes
- 36. nädala loeng 2. loeng: https://mitt.uib.no/courses/12780/files/folder/lecturenotes?preview=1065450

- Esitatud ülesanded
  - Kõik ülesanded:
    1. [N, H] põimemeetod (36. nädala loeng 2. loeng);
    2. [N, H] kiirsorteerimine (36. nädala loeng 2. loeng);
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal
    - rekursiooni vähenemine baasjuhu suunas:
      1. [N, H] põimemeetod;
      2. [N, H] kiirsorteerimine;
    - aktiveerimiskirjete ja/või rekursioonipuu joonistamine funktsiooni väljakutse põhjal:
      1. [N, H] põimemeetod;
      2. [N, H] kiirsorteerimine;
    - rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine:
      1. [N, H] põimemeetod;
- Tähelepanekuid ja häid ideid
  - Kursuseraamat kordab alguses üle rekursiivse funktsiooni definitsiooni ja reeglid.
  - Põimemeetod vs kiirsorteerimine (36. nädala loeng 2. loeng):



  - 3. nädala loengus korrati rekursiooni definitsiooni ja kasutati sama pilti, mida varasemalt õppeaine INF 101 juures (36. nädala loeng 2. loeng).

Genti Ülikool

Õppekava:
https://studiegids.ugent.be/2018/NL/FACULTY/C/BACH/CBINFO/CBINFO.html

C003770 Programming (1. semester)

- Õpiväljundid ja aine sisu https://studiegids.ugent.be/2018/NL/studiefiches/C003770.pdf:
  - Sisu: "Writing basic algorithms using recursion and loops"
- Õppeviisid:
  - "Lecture, seminar: practical PC room classes"
- Õppematerjalid:
  - Programmeren in Java met BlueJ (6de editie), Barnes & Kölling, Pearson Education, ISBN 9789043034999
  - "References"
    - Java(TM) Language Specification (Java SE 8 Edition) James Gosling, Bill Joy, Guy Steele, Gilad Bracha, Alex Buckley. Ook online raadpleegbaar.
- Tähelepanekuid ja häid ideid
  - Ingliskeelses ainekirjelduses on rekursioon, hollandikeelses ei ole.

C003773 Algorithms and Data Structures 1 (2. semester)

- Õpiväljundid ja aine sisu https://studiegids.ugent.be/2019/EN/studiefiches/C003773.pdf:
  - Sisu: "Recursion"
- Õppeviisid https://studiegids.ugent.be/2019/EN/studiefiches/C003773.pdf:
  - "Classroom lectures; Lab sessions on PC; Electronic teaching environment."
- Õppematerjalid:
  - "Course book (45 euro), Minerva-site"
  - "References:"
    - Cormen T.E., Leiserson C.E., Rivest R.L. & Stein C., "Introduction to Algorithms", MIT Press, 2005 (2nd edition).
    - Sedgewick R. & Wayne K., "Algorithms", Addison-Wesley, 2011 (4th edition).

Ulmi Ülikool

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
Ajakava: https://campusonline.uni-ulm.de/qislsf/pub/studienplan/SP-BA-Inf-FSPO2017.pdf;

8207970319 Einführung in die Informatik (1. semester)

- Õpiväljundid ja aine sisu (moodulite raamat):
  - Õpiväljundid: "elementare Strukturierungs- und Verarbeitungsmechanismen(Objektorientierung, Modularisierung, Divideand-Conquer, Iteration, Rekursion)"
  - Sisaldab: "Dynamische Datenstrukturen und ihre Verarbeitung (Listen, Bäume, Graphen,Rekursion)"
- Õppeviisid:

    - Loengud, praktikumid
- Õppematerjalid (moodulite raamat):
    - Gumm Heinz-Peter, Sommer Manfred: Einführung in die Informatik , Oldenbourg Verlag, 2006
    - Broy Manfred: Informatik - Eine grundlegende Einführung, Band 1, Programmierung und Rechnerstrukturen, Springer Verlag, 1998
    - Küchlin Wolfgang, Weber Andreas: Einführung in die Informatik - Objektorientiert mit Java, Springer Verlag, 2003
    - Echtle Klaus, Goedicke Michael: Lehrbuch der Programmierung mit Java,dpunkt Verlag, 2000

8207970318 Algorithmen und Datenstrukturen (3. semester)

- Õpiväljundid ja aine sisu (moodulite raamat):
    - Sisu: "Analyse rekursiver Algorithmen und der dabei entstehenden Rekursionsgleichungen, Mastertheorem."
- Õppeviisid:
    - Loengud, praktikumid
- Õppematerjalid (moodulite raamat):
    - U. Schöning: Algorithmik, Spektrum Verlag, Nachdruck 2011
    - T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein: Introduction to Algorithms.Second Edition. The MIT Press, 2001.

Aalborgi Ülikool

Õppekava: https://studieordninger.aau.dk/2018/8/146

Datalogiens Teoretiske Grundlag (The Theory of Computer Science) (1. semester)

- Õpieesmärgid:
    - Viden (knowledge): Rekursion og induktion
    - Rekursiv definition af funktioner
        - Rekursiv definition af mængder og simple strukturer: strenge og træer
        - Ræsonnementer om rekursivt definerede strukturer, strukturel induktion

Algoritmik og Datastrukturer 1 (Algoritmid ja andmestruktuurid) (3. semester)

- Õpieesmärgid:
    - Viden (knowledge): matematiske grundbegreber såsom rekursion, induktion, konkret og abstrakt kompleksitet
    - FÆRDIGHEDER (skills): gennemføre kompleksitets- og korrekthedsanalyse på simple algoritmer, herunder rekursive algoritmer

Syntaks og Semantik (Süntaks ja semantika) (4. semester)

- Õpieesmärgid
    - Viden (knowledge): rekursive definitioner og beregning af fikspunkter

Avancerede Algoritmer (Algoritmid edasijõudnutele) (5. semester)

- Õpieesmärgid
    - Viden (knowledge): algoritmeanalyse teknikker såsom rekursion, amortiseret analyse, analyse af forventet kompleksitet og eksperimenter med algoritmer

Modellering og Verifikation (Modelleerimine ja verifikatsioon) (6. semester)

- Õpieesmärgid
    - Viden (knowledge): "Specielt skal den studerende opnå viden om: … Hennessy-Milner logik med rekursion …"

Kataloonia Polütehniline Ülikool (KPÜ)

Õppekava: Informatics Engineering (Major in Computer Engineering or Computing or Information Systems or Information Technologies or Software Engineering)
https://www.upc.edu/en/bachelors/informatics-engineering-barcelona-fib

PRO1 Programming I (1. semester)

- Õpiväljundid ja aine sisu https://www.upc.edu/content/grau/guiadocent/pdf/ing/270001:
    - Õpiväljundid: "Understand recursion and develop recursive solutions for simple problems"
    - Sisu: "Degree competences to which the content contributes: Description: Introduction to recursive design."
- Õppeviisid:
    - Teooria ("the lecturer will alternate new theoretical concepts with examples and exercises") ja praktika ("The laboratory sessions have two distinct parts: During the first hour, a guided session takes place, where the lecturer describes practical issues regarding the programming environment, or some exercises are solven in a collaborative way, or some code is analyzed to identify errors, etc. Then students devote the remaining two hours to solve problems with the automatic judge with the assistance of the lecturer if needed.")
    - Laboratooriumimaterjalides rekursiooni ei mainitud.
    - 7. loeng: http://www.cs.upc.edu/~jordicf/Teaching/programming/pdf4/IP07_Recursion-4slides.pdf
    - 15. loeng: http://www.cs.upc.edu/~jordicf/Teaching/programming/pdf4/IP15_Conclusions-4slides.pdf
- Õppematerjalid:
    - Aine veebileht materjalidega: http://www.lsi.upc.edu/pro1/
    - Katalaanikeelne materjal, mida keelebarjääri tõttu arvesse pole võetud: http://www.lsi.upc.edu/pro1/data/uploads/apuntsip.pdf

- Professorat de Programació 1. Transparències de teoria de l'assignatura (Pàgina web de l'assignatura, apartat material docent) http://www.lsi.upc.edu/pro1/
- Professorat de Programació 1. Dotze algorismes fonamentals (Pàgina web de l'assignatura, apartat material docent) http://www.lsi.upc.edu/pro1

- Esitatud ülesanded
  - Kõik ülesanded:
    1. [HL] arvu n binaarkuju (7. loeng);
    2. [HL] Fibonacci (7. loeng);
    3. [NH] Hanoi tornid (7. loeng);
    4. [N, H] binaarotsing;
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal
    1. puurekursioon (sh binaarne rekursioon):
       1. [HL] arvu n binaarkuju;
       2. [HL] Fibonacci;
    - mõttes etteantud funktsiooni väärtustamine olenevalt sisendist:
       1. [HL] Fibonacci;
    - baasjuhu ja sammu äratundmine ning mõistmine:
       1. [HL] arvu n binaarkuju;
       2. [HL] Fibonacci;
    - rekursiooni vähenemine baasjuhu suunas:
       1. [HL] arvu n binaarkuju;
       2. [HL] Fibonacci;
- Tähelepanekuid ja häid ideid
  - Aine kirjelduses on teemade kaupa märgitud, kui palju aega on igale õpiväljundile/ teemale planeeritud.

Topic development: "Recursion"	Hours: 11h 48m Theory classes: 2h Practical classes: 0h Laboratory classes: 3h Guided activities: 0h 48m Self study: 6h
Description: Understand and assimilate the concepts covered in theory classes. Solve the problems set for this topic, available at www.jutge.org. Specific objectives: 3, 4, 9	

  - Rekursioonist (7. loeng):
    1. "Recursion can substitute iteration in program design:
       1. – Generally, recursive solutions are simpler than (or as simple as) iterative solutions.
       2. – There are some problems in which one solution is much simpler than the other.
       3. – Generally, recursive solutions are slightly less efficient than the iterative ones (if the compiler does not try to optimize the recursive calls).
       4. – There are natural recursive solutions that can be extremely inefficient. Be careful !"

- Rekursiivset disaini tutvustati kolme reegli abil (faktoriaali näitega) (7. loeng):
    1. "Identify the basic cases (those in which the subprogram can solve the problem directly without recurring to recursive calls) and determine how they are solved.
    2. Determine how to resolve the non-basic cases in terms of the basic cases, which we assume we can already solve.
    3. Make sure that the parameters of the call move closer to the basic cases at each recursive call. This should guarantee a finite sequence of recursive calls that always terminates."
- "Good programming methodology helps to create correct programs (15. loeng):
    1. Start from the specification, not from your idea.
    2. Divide a complex problem into smaller pieces (procedures and functions). Carefully specify each one.
    3. Use induction to guide your design of loops and your use of recursion. How do I solve problems of size n, if I could solve problems of size m<n?
    4. Invariants are a way to express the inductive hypothesis behind a loop."

PRO2 Programming II (2. semester)

- Õpiväljundid ja aine sisu:
    - Õpiväljundid https://www.upc.edu/content/grau/guiadocent/pdf/ing/270005:
        1. "To know the data types typically used to represent and manage linear data structures and their specification. To design iterative and recursive algorithms for solving search and traversal problems on stacks, queues and lists, using the operations of the corresponding data type and iterators (when appropriated)."
        2. "To know data types used to represent and manage tree data structures and their specification. To design recursive algorithms for solving search and traversal problems about binary, n-ary and general trees, using the operations of the corresponding data type."
        3. "To describe the main steps in the design of recursive algorithms. To justify the correctness of relatively simple recursive algorithms."
        4. "Given a simple recursive algorithm, to determine whether there is a straightforward way to obtain an equivalent iterative algorithm, and write it if there exists such a straightforward transformation."
        5. "To distinguish whether the cost of a given iterative or recursive simple algorithm which works on vectors, stacks, queues, lists or trees is linear or if it is quadratic (assuming that the cost is of one of these two types).
        6. To determine if the efficiency of a given simple recursive algorithm can be improved and, if it is possible, to design a more efficient recursive algorithm that solves the same p"
    - Sisu:
        1. "Recursive programming and correctness of recursive algorithms (6h)
            1. Inductive design of recursive algorithms. Justification of the correctness of recursive algorithms. Generalisation of a function .

Specification immersions by weakening the postcondition and strengthening the precondition. Relationship between tail-recursive algorithms and iterative algorithms.

2. Efficiency enhancements for recursive and iterative programs (6h)
    1. Detection of repeated calculations in recursive and iterative programs. Efficiency generalisations: new data (input parameters) and/or results (return values or output parameters) in recursive operations to improve efficiency. New local variables that use their previous values in iterative operations to improve efficiency.
3. Recursive data types (32h)
    1. Introduction to the use of recursive data types. Pointer type constructor and dynamic memory management. Implementation of linked data structures by means of recursive data types. Iterative and recursive algorithms for solving search and traversal problems in linked data structures by directly accessing the representation based on nodes and node pointers"

- Õppeviisid:
    - Fokusseeritud praktikale. Iganädalased loengud (2-tunnised) ja laboratooriumid (3-tunnised). Programmeerimisprojekt.
- Õppematerjalid:
    - Kursuse veebileht: http://www.cs.upc.edu/pro2/
    - Alquezar, René; Valles, Borja; [et al.]. Apunts de teoria de PRO2 [on line]. [Consultation: 20/03/2019]. Available on: <http://www.cs.upc.edu/~pro2/index.php?id=material-docent>.
    - Valles, Borja; [et al.]. Sesions de Laboratori de PRO2 [on line]. [Consultation: 20/03/2019]. Available on: <http://www.cs.upc.edu/~pro2/index.php?id=sessions>.
    - Cortadella, Jordi; Rubio, Albert; Valentin, Luis. Iniciació a la Programació (notes de curs) [on line]. 2001 [Consultation: 20/03/2019]. Available on: <http://www.cs.upc.edu/~pro2/index.php?id=material-docent>.
    - Cortadella, J.; Rubio, A.; Valentin, L. Programació 1 (notes de curs) [on line]. 2001 [Consultation: 20/03/2019]. Available on: <http://www.cs.upc.edu/pro1/>.
    - Weiss, M.A. Data structures and problem solving using C++. 2nd ed. Upper Saddle River: Pearson Education International, 2003. ISBN 0321205006.
    - Musser, D.R.; Derge, G.J.; Saini, A. STL tutorial and reference guide: C++ programming with the standard template library. 2nd ed. Boston: Addison-Wesley, 2000. ISBN 9780321702128.
    - Katalaanikeelsed materjalid ilma ülesanneteta:
        1. http://www.cs.upc.edu/pro2/data/uploads/it_rec_corr.pdf
        2. http://www.cs.upc.edu/pro2/data/uploads/efic.pdf
        3. http://www.cs.upc.edu/pro2/data/uploads/implrec.pdf
    - 8. praktikum: http://www.cs.upc.edu/pro2/data/uploads/sesion8.pdf
    - 10. praktikum: http://www.cs.upc.edu/pro2/data/uploads/sesion10.pdf
- Esitatud ülesanded

    - Kõik ülesanded:
        1. [H] 8. praktikum: pesumaja;
        2. [H] 10. praktikum: puurekursioon (sh binaarne rekursioon);
        3. [H] Apuntadores I: rekursiivse lahenduse iteratiivseks tegemine (x2);
    - Ülesannete klassifikatsioon tehniliste oskuste põhjal
        - rekursiivselt lahendatud probleemi iteratiivselt lahendamine:
            1. [H] Apuntadores I: rekursiivse lahenduse iteratiivseks tegemine (x2);
        - iteratiivse ja rekursiivse koodi ajalise keerukuse vahe:
            1. [H] 8. praktikum: pesumaja;
        - iteratiivse ja rekursiivse koodi mälukasutuse vahe:
            1. [H] 8. praktikum: pesumaja;
        - rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine:
            1. [H] 8. praktikum: pesumaja;
- Tähelepanekuid ja häid ideid
    - Äärmiselt eeskujulikud ainekirjeldusfailid.

270012 - EDA - Data Structures and Algorithmics (3. semester)

- Õpiväljundid ja aine sisu https://www.upc.edu/content/grau/guiadocent/pdf/ing/270012:
    - Õpiväljundid: "Describe the efficiency of recursive algorithms using recurrence relations and understand and apply master theorems to solve them."
    - Sisu: "Analysis of Algorithms (16h): Cost in time and space. Worst case, best case and average case. Asymptotic notation. Analysis of the cost of iterative and recursive algorithms."
- Õppeviisid:
    - Fokusseeritud praktikale. Iganädalane loeng (2h). Ülenädalased laboratooriumid (2h) ja probleemilahendamise praktikumid (2h).
    - Kasutusel C++.
    - "Programming for the game integrates knowledge and skills of the entire course."
- Õppematerjalid:
    - Kursusel ei ole kodulehte.
    - Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. Introduction to algorithms. 3rd ed. Cambridge: MIT Press, 2009. ISBN 9780262033848.
    - Brassard, G.; Bratley, P. Fundamentos de algoritmia. Madrid: Prentice Hall, 1997. ISBN 848966000X.
    - Weiss, M.A. Data structures and algorithm analysis in C++. 4th ed int. Boston: Pearson, 2014. ISBN 0273769383.
    - Kreher, D.L.; Stinson, D.R. Combinatorial algorithms: generation, enumeration and search. Boca Raton: CRC Press, 1999. ISBN 084933988X.
    - Garey, M.R; Johnson, D.S. Computers and intractability: a guide to the theory of NP-Completeness. New York, NY: W.H. Freeman, 1979. ISBN 0716710447.
    - Stroustrup, B. The C++ programming language. 4th ed. Upper Saddle River, NJ: Addison-Wesley, 2013. ISBN 9780321563842.

- Neapolitan, R.E. Foundations of algorithms. 5th ed. Burlington, MA: Jones and Bartlett Learning, 2015. ISBN 9781284049190

Helsinki Ülikool

Õppekava: https://www.helsinki.fi/fi/ohjelmat/kandi/tietojenkasittelytieteen-kandiohjelma/opintojen-rakenne-ja-aikataulu

TKT20001 Tietorakenteet ja algoritmit (Datastructures and Algorithms) (2. semester)

- Õpiväljundid ja aine sisu https://courses.helsinki.fi/en/tkt20001:
  - Sisu: "Opintojakson tarkempaa sisältöä päivitetään tarpeen mukaan. Keskeisiä aihealueita ovat esim." "algoritmien suunnittelun ja analyysin perustekniikoita: rekursio, silmukkainvariantti, iso-O-merkintä, iteratiivisten ja rekursiivisten algoritmien aika- ja tilavaativuus" st „Basic Techniques for Designing and Analyzing Algorithms: Recursion, Loop Variant, Big O, Time and Space Requirements for Iterative and Recursive Algorithms"
  - Eeldused: "Esitietovaatimuksena on Ohjelmoinnin perus- ja jatkokurssi sekä Johdatus yliopistomatematiikkaan, tai vastaavat tiedot." st „"The prerequisite requirement is the Basic and Further Programming in Programming and Introduction to University Mathematics, or similar information."
- Õppeviisid:
  - Loengud, praktikumid, iseseisev töö.
- Õppematerjalid:
  - Kursus baasneb loengumaterjalidel, mida ülikool on ise valmistanud: https://www.cs.helsinki.fi/u/ahslaaks/tirakirja/
    Raamat: Antti Laaksonen, Data Structures and Algorithms Book
  - Soovitus: Cormen, Leiserson, Rivest, Stein: Introduction to Algorithms
  - Moodle (sisenetav külalisena): https://moodle.helsinki.fi/course/view.php?id=32099
  - Kombineeritud loenguslaidid: https://moodle.helsinki.fi/mod/resource/view.php?id=1494225
- Esitatud ülesanded
  - Kõik ülesanded:
    1. [N, HL] astendamine;
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal
    - mõttes etteantud funktsiooni väärtustamine olenevalt sisendist:
      1. [HL] astendamine;
- Tähelepanekuid ja häid ideid
  - Tõid loenguslaides välja järgneva "Esittelemme ensin rekursion perusidean yksinkertaisilla "leikkikaluesimerkeillä" ja tarkastelemme sitten rekursiota hieman realistisemmissa sovelluksissa" st. esitame esiteks rekursiooni põhiideed lihtsate "mängumärkidega" ja seejärel vaatame rekursiooni realistlikumates rakendustes.

- Selgitavad, et astendamine ei ole tegelikult hea näide rekursiooniks, sest iteratiivne implementatsioon on kiirem ja kasutab vähem mälu.
- Loenguslaididel on väga vähe küsimusi publikule! Näidete arv > harjutamise arv ei ole ainult probleem UTs.
- Rekursiooni definitsioon on kohe UT kursuse Algoritmid ja Andmestruktuurid võrdses kursuses.

Chalmersi Ülikool

Õppekava: http://www.cse.chalmers.se/DV/UtbildningsplanG2018-270_N1COS.pdf
Kõikide kursuste otsing: https://gul.gu.se/public/courseId/38407/coursePath/38212/ecp/lang-sv/listUnderlyingEvents.do

X TDA555/ DIT440 Introduktion till funktionell programmering (Introduction to Functional Programming) (1. semester)

https://gul.gu.se/public/courseId/85058/coursePath/38212/38407/40148/ecp/lang-sv/publicPage.do?item=41220902

- Õpiväljundid ja aine sisu http://kursplaner.gu.se/pdf/kurs/en/DIT440:
  - Õpiväljund: "describe a basic repertoire of functional programming techniques, such as: recursion…"
  - Sisu: "recursion and recursive data types"
- Õppeviisid:
  - loengud (2/ ndl), grupikoosolekud ja praktikumid (1/ ndl).
- Õppematerjalid:
  - Kursuse veebileht: http://www.cse.chalmers.se/edu/year/2018/course/TDA555/index.html
  - Kõik loengu- ja lisamaterjalid: http://www.cse.chalmers.se/edu/year/2018/course/TDA555/lectures.html
  - 2. nädala harjutused: http://www.cse.chalmers.se/edu/year/2018/course/TDA555/ex-week2.html
  - 1. loeng: http://www.cse.chalmers.se/edu/year/2018/course/TDA555/Material/Recursion/slides.pdf
  - 2. loeng: http://www.cse.chalmers.se/edu/year/2018/course/TDA555/Material/DataTypes1/slides-2015.pdf
  - Konkreetset ühte raamatut ei ole, aga materjalide all viidatakse järgnevale:
    1. http://learnyouahaskell.com/
    2. Loenguslaidid: http://www.cse.chalmers.se/edu/year/2018/course/TDA555/schedule.html
  - Youtube videod:
    1. https://www.youtube.com/watch?v=_VLjqFM2wro&feature=youtu.be
  - Välised ülesanded:

1. https://wiki.haskell.org/H-99:_Ninety-Nine_Haskell_Problems (1-28 on kohustuslikud);

- Esitatud ülesanded
  - Kõik ülesanded:
    1. [N, HV] astendamine (1. loeng, 1. lab);
    2. [HL] n joone poolt tekitatud regioonide arv (1. loeng);
    3. [HV] arvude 1 kuni n ruutude summa (Ex W2);
    4. [H] Hanoi tornid;
    5. [H] Fibonacci;
    6. [HV] väikseim tegur (2. nädala harjutused);
    7. [H] järjendi elementide kokku korrutamine;
    8. [H] kas järjendis on duplikaate?;
    9. [H] Blackjack kaardimängu implementeerimine (2. lab);
    10. [H] n-tasemelise järjendi ühetasemeliseks muutmine (99 haskelli harj.);
    11. [H] vigase rekursiooni äratundmine (10. loeng);
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal
    - koodi lihtsustamine kordust vältides:
      1. [H] Hanoi tornid;
    - mitmepoolne rekursioon:
      1. [HV] väikseim tegur;
    - mõttes etteantud funktsiooni väärtustamine olenevalt sisendist:
      1. [HL] faktoriaal;
      2. [H] Fibonacci;
    - baasjuhu ja sammu äratundmine ning mõistmine:
      1. [HL] n joone poolt tekitatud regioonide arv;
      2. [HL] kahe naturaalarvulised astmed;
      3. [HV] astendamine;
      4. [HV] arvude 1 kuni n ruutude summa;
      5. [H] järjendi elementide kokku korrutamine;
      6. [H] kas järjendis on duplikaate?;
      7. [H] Blackjack kaardimängu implementeerimine;
      8. [H] n-tasemelise järjendi ühetasemeliseks muutmine (99 haskelli harj.);
      9. [H] vigase rekursiooni äratundmine;
    - alamülesandest terve ülesande lahenduseni jõudmine:
      1. [H] Blackjack kaardimängu implementeerimine;
    - rekursiooni vähenemine baasjuhu suunas:
      1. [HL] n joone poolt tekitatud regioonide arv;
      2. [H] kas järjendis on duplikaate?;
    - funktsiooni käitumine ootamatute argumentide korral:
      1. [H] järjendi elementide kokku korrutamine (tühi järjend);
    - algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
      1. [HV] arvude 1 kuni n ruutude summa;
      2. [H] Fibonacci;

3. [H] Blackjack kaardimängu implementeerimine;
- fikseerimata tasemete arvuga andmestruktuuride töötlemine:
  1. [H] n-tasemelise järjendi ühetasemeliseks muutmine (99 Haskelli harjutust);
- rekursiooni ajaline keerukus;
  1. [H] Fibonacci;

- Tähelepanekuid ja häid ideid
  - Üks näide ja siis alles definitsioon.
  - Kuidas rekursioon toimib:
    1. "Reduce a problem (e.g. power x n) to a smaller problem of the same kind
    2. So that we eventually reach a "smallest" base case
    3. Solve base case separately
    4. Build up solutions from smaller solutions"
  - Praktikumi ülesannete osa on enda funktsioonide testimine (nt. http://www.cse.chalmers.se/edu/year/2018/course/TDA555/lab1.html)
  - Loenguslaidide lõpus on osa "What Did We Learn?", mille abil tudeng saab enda õpitule kriitiliselt otsa vaadata (2. loeng).
  - Palju seoses funktsionaalse programmeerimisega.

X DIT961 Datastrukturer (Data Structures) (4. semester)

- Õpiväljundid ja aine sisu: kursuse veebilehekülgelt
  - Kursuse plaan: "Big-Oh complexity and how to reason about it, in iterative and recursive programs"
- Õppeviisid:
  - loengud, harjutused, praktikumid.
- Õppematerjalid:
  - Ametlikku raamatut ei ole.
  - Kursuse veebilehekülg: http://www.cse.chalmers.se/edu/year/2018/course/DIT961/
  - 4. ja 6. nädala harjutused: http://www.cse.chalmers.se/edu/year/2018/course/DIT961/selfstudy/;
  - Äge viis algoritme visualiseerida: https://visualgo.net/en/list;
  - Algoritmide visualiseerimiseks: https://www.cs.usfca.edu/~galles/visualization/Algorithms.html
  - 3. loeng: http://www.cse.chalmers.se/edu/year/2018/course/DIT961/files/lectures/dit961_lecture_3.pdf;
  - 5. loeng: http://www.cse.chalmers.se/edu/year/2018/course/DIT961/files/lectures/dit961_lecture_5.pdf;
  - 8. loeng: http://www.cse.chalmers.se/edu/year/2018/course/DIT961/files/lectures/dit961_lecture_8.pdf;

    - 11. loeng: http://www.cse.chalmers.se/edu/year/2018/course/DIT961/files/lectures/dit961_lecture_11.pdf;
- Esitatud ülesanded
    - Kõik ülesanded:
        1. [H] sabarekursiooni äratundmine (11. loeng);
        2. [H] järjendi alamjärjendid (4. nädala harjutused);
        3. [H] binaarpuu elemendid kasvavas järjekorras (6. nädala harjutused);
    - Ülesannete klassifikatsioon tehniliste oskuste põhjal
        - puurekursioon (sh binaarne rekursioon):
            1. [H] binaarpuu elemendid kasvavas järjekorras;
        - lineaarne- ja sabarekursioon:
            1. [H] sabarekursiooni äratundmine;
        - baasjuhu ja sammu äratundmine ning mõistmine:
            1. [H] järjendi alamjärjendid;
        - rekursiooni mälukasutus;
            1. [N] kahekordne sabarekursioon;
        - rekursiooni ajaline keerukus;
            1. [H] järjendi alamjärjendid;
- Tähelepanekuid ja häid ideid
    - Rekursiooni definitsiooni/ olemust ei korrata aine alguses üle, minnakse kohe rekursiooni keerukuse arvutamise juurde. (3. loeng)
        1. “Procedure for calculating complexity of a recursive algorithm: •
            1. Write down a recurrence relation e.g. T(n) = O(n) + 2T(n/2)
            2. Solve the recurrence relation to get a formula for T(n) (difficult!)
            3. There isn't a general way of solving any recurrence relation – we'll just see a few families of them”
    - Korratakse üle alles 8. nädalal (11. loeng)
    - Erinevatest algoritmidest: divide and conquer, quick sort, partitioning, mergesort. Rekursiooni sobivusest/ keerukuseks võrreldes iteratiivse lahendusega ei räägita. (5. loeng)
    - Naive merging, leftist merging, skew merging. Rekursiooni sobivusest/ keerukuseks võrreldes iteratiivse lahendusega ei räägita. (8. loeng)
    - Samuti funktsionaalsetele keeltele keskendub (Haskell).

Aalto Ülikool

Üldine kursusematerjalide otsing: https://mycourses.aalto.fi/course/index.php?categoryid=35
Õppekava: https://into.aalto.fi/display/enbsctech/Data+Science

CS-A1110 Programming 1 (1. semester)

- Õpiväljundid ja aine sisu https://oodi.aalto.fi/a/opintjakstied.jsp?html=1&Tunniste=CS-A1110&Kieli=6:

- Sisu: Additional content: ..., recursion ....”
- Mida veel õppida saab https://plus.cs.hut.fi/o1/2018/wNN/goals/?
    1. “You understand **the concept of recursion** and have seen examples of recursively defined data and recursive method implementations. You can implement a simple recursive class or method.”
- Teemad e-raamatu põhjal:
    1. “Recursively defined (self-referential) data.
    2. Recursion as an implementation technique for functions.
    3. Some aspects of algorithm efficiency.”

- Õppematerjalid:
    - Kursuse veebileht: https://plus.cs.hut.fi/o1/2018/
    - e-õpiku 11. peatükk: https://plus.cs.hut.fi/o1/2018/w11/ch02/
    - Loenguid **ei** ole! Kasutatakse ainult e-raamatut ja praktikume.
    - e-õpikus:
        1. animatsioonid;
        2. interaktiivsed küsimused (esitamisvormiga);
- Esitatud ülesanded
    - Kõik ülesanded:
        1. [HV] faktoriaal (e-õpiku 11. peatükk);
        2. [HV] maks, mida tuleb saadud kingituse pealt maksta (e-õpiku 11. peatükk);
        3. [H] Viinaharava mängu täiendamine (e-õpiku 11. peatükk);
        4. [N, H] lemmikute asjade hoidmisstruktuur (e-õpiku 11. peatükk);
        5. [H] sugupuu (e-õpiku 11. peatükk);
        6. [H] Fibonacci (e-õpiku 11. peatükk);
    - Ülesannete klassifikatsioon tehniliste oskuste põhjal
        - mõttes etteantud funktsiooni väärtustamine olenevalt sisendist:
            1. [H] täisarvude 1 kuni n ruudud (valikvastustega) (x2, iteratiivse ja rekursiivse vahe sama sisendi korral);
            2. [H] lemmikute asjade hoidmisstruktuur;
            3. [H] Fibonacci;
        - baasjuhu ja sammu äratundmine ning mõistmine:
            1. [H] sugupuu;
        - rekursiooni vähenemine baasjuhu suunas:
            1. [H] Viinaharava mängu täiendamine;
            2. [H] sugupuu;
        - koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine:
            1. [HV] faktoriaal;
            2. [HV] maks, mida tuleb saadud kingituse pealt maksta;
            3. [H] Viinaharava mängu täiendamine (olemasolev oli iteratiivne!);
        - algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
            1. [HV] maks, mida tuleb saadud kingituse pealt maksta;
        - rekursiooni mälukasutus;
            1. [H] Fibonacci;

- Tähelepanekuid ja häid ideid
  - Rekursioon 11. nädalal (kokku 13).
  - Funktsionaalse keele Scala põhjal.
  - Loenguid **ei** ole (va 3 organisatoorset)! https://plus.cs.hut.fi/o1/2018/wNN/lectures/
  - Väga hea erialasõnastik viidetega e-raamatule: https://plus.cs.hut.fi/o1/2018/wNN/glossary/
  - Sarnane Programmeerimine kursusele. Samas käsitleb natukene ka algoritmiefektiivsust.
  - "*Rough Estimate of Workload:*? Around four hours."
  - Alustab rekursioonist just andmetüüpe arvestades ja OOP kontekstis. Defineerib strukturaalse rekursiivsuse (e-õpiku 11. peatükk):
    1. "In a structurally recursive object-oriented program, objects contain references to other objects of the same type: the objects can thus form a chain of prerequisites, a company's personnel tree, a network of game areas, or a similar linked structure."
  - Tabel, mis illustreerib iteratiivsete ja rekursiivsete algoritmide tunnuseid/ vahe (e-õpiku 11. peatükk):

	Iteration with loops	Recursion
Advancing the algorithm:	We step through the problem by repeating an operation. Each iterative cycle continues where the previous one left off. Example: we repeatedly check pairs of letters before moving to the next pair within the candidate palindrome. Example: we loop through a chain of courses, repeatedly checking the credit values and adding them to an accumulating sum.	We chip off a piece of the problem that is easy to solve and apply the same algorithm to the reduced problem. . Example: we remove the first and last characters (a piece of the problem that is easily dealt with) and apply the same algorithm to the remaining middle part (the reduced problem). Example: we take the target course's own credit value (a piece of the problem) and apply the same algorithm to its prerequisites (the reduced problem).
Use of variables:	We use `vars` and mutable collections for tracking the algorithm's state. Example: we use two stepper variables to track the two indices within the candidate word. Example: We use a `var` to track which of the chained courses we're currently working on. We modify that `var` on each iterative cycle. Another `var` serves as a gatherer where we accumulate the sum as we loop through the courses.	We can get the job done using only parameters and other `val` variables. Example: we pass the remaining (middle) part of the string as a parameter to the next frame, higher on the stack. Example: `this` points to a different course in each stack frame. The intermediate sums are produced and returned by different frames; we don't need to modify the value of any variable.
Termination:	We stop repeating when we have advanced through the entire problem. Example: we exit the loop when the increasing "left index" meets the decreasing "right index". Example: we loop until we hit a course that has no prerequisite.	When we've chipped away enough that only a little crumb of the original problem (a base case) remains, the solution is obvious and doesn't need another recursive call. Example: when we have a string of at most one character, the solution is obvious. Example: if the course is an intro course, its total credits are obvious.
Constructing the solution:	Once we've gone through the entire problem, we have accumulated a solution. Example: by the time we've checked the whole string, we know if it's a palindrome. Example: we track the sum in a gatherer variable, adding to it at each step. Once we're done, the variable contains the final result.	We form an overall solution from the solution of the chipped-away piece and the solution of the reduced problem. Often, the overall solution will be some kind of combination of the two (such as a sum). Example: if the first and last characters are different (solution of chipped-away piece), the string is not a palindrome. Otherwise, the solution equals that of the remaining middle part (the reduced problem). Example: the "total credits" of a course is the sum of its own credit value (solution of chipped-away piece) and the total credits of the prerequisites (the reduced problem).

  - Enne rekursiivse funktsiooni kirjutamist kirjutatakse pseudokood (e-õpiku 11. peatükk).

- Toob välja ebaefektiivse Fibonacci ja pakub välja ka koodi palju efektiivsema Fibonacciga (e-õpiku 11. peatükk).
- e-raamatu peatüki lõpus on "Summary of Key Points" ja võimalus anda tagasisidet materjali headuse kohta. (ajakulu, asjadest arusaam, kirjalik tagasiside) (e-õpiku 11. peatükk).

CS-A1120 Programming 2 (2. semester)

https://mycourses.aalto.fi/course/view.php?id=20545

- Õpiväljundid ja aine sisu: https://oodi.aalto.fi/a/opintjakstied.jsp?html=1&Tunniste=CS-A1120&Kieli=6:
  - Õpiväljundid:
    1. "(v) understands the concept of recursion and can write programs that employ recursion and operate on recursive data structures, and"
    2. Moodulite õpieesmärkidest:
       1. "recall from CS-A1110 Programming 1 ("Ohjelmointi 1")… how to work with recursion in Scala"
       2. "In this round you will learn …
       3. … that recursion is one of the basic programming principles to organize computations and data
       4. … that the power of recursion stems from controlled self-reference that terminates at base cases
       5. in essence, we can implement a large computation (or data structure) using smaller, self-similar, parts until the computation is so small so as to become trivial
       6. … that many familiar mathematical objects and functions benefit from a recursive definition
          1. for example, a *string* of *length* nn either (i) is empty (when n=0n=0, the base case), or (ii) consists of a first *character* followed by a *string* of *length* n−1n−1 (when n≥1n≥1, the recursive case)
          2. recursive definitions lead to simple recursive functions that manipulate data or carry out a computation
       7. … that recursion is naturally associated with a recursion tree that records all the stages of recursion
       8. … that recursion is a natural tool to carry out exhaustive search
       9. … how to use tail recursion to obtain efficiency"
  - Sisu:
    "Aspects of functional programming, e.g. recursive definitions and recursion."
- Õppeviisid:
  - "Lecture notes, articles, and programming assignments"
- Õppematerjalid:
  - Kursuse üldinfo: https://mycourses.aalto.fi/course/view.php?id=20545
  - Kursuse veebileht: http://a1120.cs.aalto.fi/notes/

- e-õpiku peatükk „Recursion and tail-recursion recalled“: http://a1120.cs.aalto.fi/notes/round-recursion--recursion.html
- e-õpiku peatükk „Recursively defined data structures“: http://a1120.cs.aalto.fi/notes/round-recursion--rdds.html
- e-õpiku peatükk “Linked lists”: http://a1120.cs.aalto.fi/notes/round-recursion--lists.html
- e-õpiku peatükk „Solving problems recursively“: http://a1120.cs.aalto.fi/notes/round-recursion--solving.html
- **NB!** Ühelegi ülesandele juurdepääsu ei ole!

- Esitatud ülesanded
  - Kõik **näited (**näiteid ei võetud arvesse ülesannete arvu ja tehniliste oskuste statistikas):
    1. [N] faktoriaal (e-õpiku peatükk „*Recursion and tail-recursion recalled*“);
    2. [N] positiivsete täisarvude summa arvuni n (e-õpiku peatükk „*Solving problems recursively*“);
    3. [N] palindroom (e-õpiku peatükk „*Recursion and tail-recursion recalled*“);
    4. [N] kas lihtahel sisaldab elementi e (e-õpiku peatükk “*Linked* lists”);
    5. [N] järjendi elementide arv (e-õpiku peatükk “*Linked lists*”);
    6. [N] hulga alamhulk, mille summa võrdub x’iga (e-õpiku peatükk „*Solving problems recursively*“).
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal
    - lineaarne- ja sabarekursioon:
      1. [N] faktoriaal;
      2. [N] kas järjend sisaldab elementi e;
      3. [N] kas järjend sisaldab elementi e;
    - mõttes etteantud funktsiooni väärtustamine olenevalt sisendist:
      1. [N] faktoriaal;
      2. [N] positiivsete täisarvude summa arvuni n;
    - koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine:
      1. [N] palindroom;
      2. [N, H] kas järjend sisaldab elementi e;
      3. [N] hulga alamhulk, mille summa võrdub x’iga;
    - algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
      1. [N] faktoriaal;
    - rekursiooni mälukasutus;
      1. [N] faktoriaal;
    - rekursiooni ajaline keerukus;
      1. [N] palindroom;
    - iteratiivse ja rekursiivse koodi ajalise keerukuse vahe:
      1. [N] faktoriaal;
    - iteratiivse ja rekursiivse koodi mälukasutuse vahe:
      1. [N] kas järjend sisaldab elementi e (mitte sabarekursiivne vs iteratiivne vs sabarekusriivne vs Scala sisseehitatud versioonid)

        - aktiveerimiskirjete ja/või rekursioonipuu joonistamine funktsiooni väljakutse põhjal:
            1. [N] palindroom;
            2. [N] faktoriaal;
            3. [N] kas järjend sisaldab elementi e;
- Tähelepanekuid ja häid ideid
    - Aine jagatud kolmeks mooduliks (+ soojendusraund). Igal moodulil omakorda õpieesmärgid.
    - Soojendusraundis korrati rekursioon üle kahe lihtsa näite põhjal.
    - Olgugi, et kordab veel põhimõtte üle, siis viitas tagasi eelneva kursuse rekursiooni materjalile.
    - Omaette teema on rekursiivselt defineeritud andmestruktuurid: *symbolic arithmetic expressions.* linked lists ja expressions (e-õpiku peatükk „Recursively defined data structures“);
    - “Not only functions can be defined recursively but also data structures. Usually, such data structures are also most naturally manipulated with recursive functions.”
        - “Linked lists. This is a classic data structure and corresponds to the List class in Scala.
        - Symbolic arithmetic expressions. This is an instance of a very common tree data structure.” (e-õpiku peatükk „*Recursively defined data structures*“);
    - Eraldi raund “Solving problems recursively” (e-õpiku peatükk „Solving problems recursively“):
        1. “Recursion is a natural approach to implement an *exhaustive search* that considers, one step at a time, all possibilities for arriving at a *solution*. As such, recursion is a natural tool for solving *search problems*.”
        2. Probleem: subset sum problem. Ei ole kõige efektiivsem, aga paremat algoritmi samuti veel ei ole “*Currently, no one knows whether this is possible. No such algorithm is known.”*

CS-A1140 Data Structures and Algorithms (3. semester)

- Õpiväljundid ja aine sisu https://mycourses.aalto.fi/course/view.php?id=20548&section=2:
    - Rekursiooni ei mainita.
- Õppematerjalid:
    - Õpik: Introduction to Algorithms By Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest
    - 8. loeng: http://a1120.cs.aalto.fi/slides/lecture8.pdf
    - Muudele õppematerjalidele ei ole ligipääsu

CS-E3190 Principles of Algorithmic Techniques (5. semester)

Põnev materjal AA täiendamiseks.
https://mycourses.aalto.fi/course/view.php?id=16917#section-0

Materjali on küllaga (loenguslaidid, praktikumimaterjalid). Rekursiooni loomulikult mainitakse.

- Õppematerjalid:
  - 4. loeng: https://mycourses.aalto.fi/pluginfile.php/516418/mod_resource/content/7/CS-E3190_lect04.pdf
- Tähelepanekuid:
  - "Binary search can (and should) be implemented easily also without recursion." (4. loeng)
  - Rekursiooniga tegeletakse ainult ajalise keerukusega kontekstis.

Marylandi Ülikool, College Park

- Nelja aasta plaanide näited: https://undergrad.cs.umd.edu/future
- Kõik "Class Web Pages": http://www.cs.umd.edu/class/

CMSC 131A Object-Oriented Programming I (1. semester)

- Õpiväljundid ja aine sisu http://www.cs.umd.edu/class/fall2017/cmsc131A/Syllabus.html:
  - Sisu: Generative recursion, …, recursion with accumulators
- Õppeviisid:
  - Loengud, klikkeriküsimused, praktikumid
- Õppematerjalid:
  - Kursuseleht: http://www.cs.umd.edu/class/fall2017/cmsc131A/Syllabus.html
  - How to Design Programs, Second Edition (On-line Draft) by Felleisen, Findler, Flatt, Krishnamurthi. MIT Press.
  - Märkmed (*notes)*: http://www.cs.umd.edu/class/fall2017/cmsc131A/Notes.html
  - Laboratooriumid:
    1. Fraktaalides midagi 20. lab: http://www.cs.umd.edu/class/fall2017/cmsc131A/lab20.html
    2. http://www.cs.umd.edu/class/fall2017/cmsc131A/lab21.html
    3. http://www.cs.umd.edu/class/fall2017/cmsc131A/lab22.html
    4. http://www.cs.umd.edu/class/fall2017/cmsc131A/lab23.html
    5. http://www.cs.umd.edu/class/fall2017/cmsc131A/lab25.html
- Esitatud ülesanded
  - Kõik ülesanded:
    1. [N, H] faktoriaal;
    2. [N, H] mittetühja reaalarvude järjendi suurim element;
    3. [N, H] järjendi elementide korrutis;
    4. [N, H] järjendi elementide summa;
    5. [HV] kas etteantud tähestikulises järjekorras kasvav string (AscSring) sisaldab tähte 1String?;
    6. [H] kilpkonnaga joonte joonistamine;

7. [HL] fraktaal;
8. [H] vähenevate ringide joonistamine;
9. [HV] vähenevate ringide joonistamine spiraalina;
10. [HV] puufraktaal;
11. [H] n-is Fibonacci arv;
12. [H] kõik Fibonacci arvud, mis on väiksemad, kui n;
13. [H] kas naturaalarv n on algarv?;
14. [H] kõik algarvud <=n;
15. [H] Ackermanni funktsioon (mitmeetapiline funktsioon);
16. [H] Gaussi tehnika: arvude 1 kuni naturaalarv n kokku liitmine;
17. [H] kartesiaanlik korrutis;
18. [HV] järjend kõikidest puudest kõrgusega h;

- Ülesannete klassifikatsioon tehniliste oskuste põhjal
  - rekursiivselt lahendatud probleemi iteratiivselt lahendamine:
    1. [H] kartesiaanlik korrutis (peale rekursiivset lahendust);
  - baasjuhu ja sammu äratundmine ning mõistmine:
    1. [HV] kas etteantud tähestikulises järjekorras kasvav string (AscSring) sisaldab tähte 1String?;
    2. [H] kilpkonnaga joonte joonistamine;
    3. [HL] fraktaal;
    4. [H] vähenevate ringide joonistamine;
    5. [HV] vähenevate ringide joonistamine spiraalina;
    6. [HV] puufraktaal;
    7. [H] fraktaal – Kochi lumehelves;
    8. [H] n-is Fibonacci arv;
    9. [H] kõik Fibonacci arvud, mis on väiksemad, kui n;
    10. [H] kas naturaalarv n on algarv?;
    11. [H] kõik algarvud <=n;
    12. [H] Ackermanni funktsioon (mitmeetapiline funktsioon);
    13. [H] Gaussi tehnika: arvude 1 kuni naturaalarv n kokku liitmine;
    14. [H] kartesiaanlik korrutis;
    15. [HV] järjend kõikidest puudest kõrgusega h;
    16. [HV] kõikide h-kõrguseliste puude arv;
    17. [HV] kõikide h-kõrguseliste puude arv;
  - alamülesandest terve ülesande lahenduseni jõudmine:
    1. [H] kilpkonnaga joonte joonistamine;
    2. [H] vähenevate ringide joonistamine;
    3. [HV] vähenevate ringide joonistamine spiraalina;
    4. [HV] puufraktaal;
    5. [H] fraktaal – Kochi lumehelves;
    6. [HV] järjend kõikidest puudest;
  - rekursiooni vähenemine baasjuhu suunas:
    1. [HV] kas etteantud tähestikulises järjekorras kasvav string (AscSring) sisaldab tähte 1String?;

2. [H] kilpkonnaga joonte joonistamine;
3. [H] vähenevate ringide joonistamine;
4. [HV] vähenevate ringide joonistamine spiraalina;
5. [HV] puufraktaal;
6. [H] fraktaal – Kochi lumehelves;
7. [H] Ackermanni funktsioon (mitmeetapiline funktsioon);
8. [HV] järjend kõikidest puudest kõrgusega h;

- funktsiooni käitumine ootamatute argumentide korral:
  1. [HL] fraktaal;
  2. [HV] Ackermanni funktsioon (mitmeetapiline funktsioon);
  3. [H] kartesiaanlik korrutis;
- rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast:
  1. [HL] Gaussi tehnika: arvude 1 kuni naturaalarv n kokku liitmine;
- koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine:
  1. [H] faktoriaal;
  2. [H] mittetühja reaalarvude järjendi suurim element;
  3. [H] järjendi elementide korrutis;
  4. [H] järjendi elementide summa;
  5. [H] vähenevate ringide joonistamine (eelneva sama ülesande lahenduse täiendamine);
  6. [HV] vähenevate ringide joonistamine spiraalina (eelneva sama ülesande lahenduse täiendamine);
  7. [H] kas naturaalarv n on algarv?
  8. [N] Gaussi tehnika: arvude 1 kuni naturaalarv n kokku liitmine;
  9. [H] n-is Fibonacci arv;
- algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
  1. [H] vähenevate ringide joonistamine (x2);
  2. [HV] vähenevate ringide joonistamine spiraalina;
  3. [HV] puufraktaal;
  4. [H] fraktaal – Kochi lumehelves;
  5. [H] Ackermanni funktsioon (mitmeetapiline funktsioon);
- rekursiooni mälukasutus;
  1. [H] Gaussi tehnika: arvude 1 kuni naturaalarv n kokku liitmine;
  2. [H] Gaussi tehnika: arvude 1 kuni naturaalarv n kokku liitmine (parandatud teine versioon);
  3. [H] kartesiaanlik korrutis;
  4. [HV] järjend kõikidest puudest kõrgusega h;
- rekursiooni ajaline keerukus;
  1. [HV] kas etteantud tähestikulises järjekorras kasvav string (AscSring) sisaldab tähte 1String?;
  2. [H] Gaussi tehnika: arvude 1 kuni naturaalarv n kokku liitmine;
  3. [H] Gaussi tehnika: arvude 1 kuni naturaalarv n kokku liitmine (parandatud teine versioon);
  4. [H] n-is Fibonacci arv (parandatud versioon, võrdle versioone);

        - aktiveerimiskirjete ja/või rekursioonipuu koostamine
            1. [HL] Gaussi tehnika: arvude 1 kuni naturaalarv n kokku liitmine;
        - rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine:
            1. [H] n-is Fibonacci arv;
            2. [H] kõik Fibonacci arvud, mis on väiksemad, kui n;
            3. [H] kas naturaalarv n on algarv?;
            4. [H] kõik algarvud <=n;
- Tähelepanekuid ja häid ideid
    - Funktsionaalne keel jälle esimeseks kokkupuuteks!
    - "Laptops will not be permitted in class."
    - Mainib järgmiseid kontsepte: accumulators, generative recursion.
    - Harjutuseks on enda rekursiivse koodi testimine ja eelnevate versioonidega võrdlemine: "Run a bunch of tests to see how large an input causes sum-all to run out of memory. Try to get sum-all-down to run out of memory too."

CMSC 132 Object-Oriented Programming II (2. semester)

- Õpiväljundid ja aine sisu: http://www.cs.umd.edu/class/spring2019/cmsc132-010X-030X/syllabus.pdf
  https://www.cs.umd.edu/class/summer2018/cmsc132/syllabus.shtml
    - Sisu: "Recursion (Ch. 5)" 2 loengut.
- Õppeviisid:
    - loengud, praktikumid, projektid
- Õppematerjalid:
    - Data Structures and Algorithms in Java, 6th Edition, Goodrich, Tamassia, & Goldwasser, Wiley, January 2014, ISBN 978–1–118–77133–4. A digital edition – ISBN 978–1–118–80314–1.
    - Kursuse koduleht: https://www.cs.umd.edu/class/summer2018/cmsc132/schedule.shtml
    - 10. loeng: https://www.cs.umd.edu/class/summer2018/cmsc132/lectures/10-recursion.pdf
    - 11. loeng: https://www.cs.umd.edu/class/summer2018/cmsc132/lectures/lecture11.pdf
    - Projektid: https://www.cs.umd.edu/class/summer2018/cmsc132/projects/p5/Recursion.html https://www.cs.umd.edu/class/summer2018/cmsc132/projects/p6/PolymorphicBST.html
    - Kodutöö rekursiooniga (viimane kodutöö): http://www.cs.umd.edu/class/spring2019/cmsc132-020X-040X/Project7/Bst.html
- Esitatud ülesanded
    - Kõik ülesanded:
        1. [H] rekursiivse funktsiooni väärtuse ennustamine (x2);
        2. [HL] Hanoi tornid;
    - Ülesannete klassifikatsioon tehniliste oskuste põhjal

        - mõttes etteantud funktsiooni väärtustamine olenevalt sisendist:
            1. [H] rekursiivse funktsiooni väärtuse ennustamine (x2);
        - aktiveerimiskirjete ja/või rekursioonipuu joonistamine funktsiooni väljakutse põhjal:
            1. [HL] Hanoi tornid;
- Tähelepanekuid ja häid ideid
    - Terve 10. loeng rekursioonile. Rekursiooni mõiste defineeritakse loengus uuesti.

CMSC 330 Organization of Programming Languages (4. semester)

- Õpiväljundid ja aine sisu:
    - Sabarekursioon
    - Sisu: "Lists and recursion"
- Õppeviisid:
    - loengud, projektid
    - keel: OCaml
- Õppematerjalid:
    - Kursuse veebileht: http://www.cs.umd.edu/class/spring2018/cmsc330/#home
    - 11. loeng: http://www.cs.umd.edu/class/spring2018/cmsc330/lectures/11-tailrecursion.pdf;
    - Projekt „2A: OCaml Warmup": https://github.com/anwarmamat/cmsc330spring18-public/tree/master/p2a.
- Esitatud ülesanded
    - Kõik ülesanded:
        1. [HL] kas funktsioon *map* on sabarekursiivne? (11. loeng);
        2. [HL] kas funktsioon *fold* on sabarekursiivne? (11. loeng);
        3. [HL] kas funktsioon *fold_right* on sabarekursiivne? (11. loeng);
        4. [HL] koodi põhjal sabarekursiivsuse tuvastamine (map) (11. loeng);
        5. [HL] sabarekursiivse *map*i ajaline keerukus (11. loeng);
        6. [H] Eukleidese algoritm – suurim ühistegur (projekt);
        7. [H] Ackermanni funktsioon (projekt);
        8. [H] järjendi elementide summa kuni argumendina antud indeksini (projekt);
        9. [H] järjendi elementide summa kuni argumendina antud indeksijärjendini ([0; 1] [5;6;7] -> [5; 11]) (projekt);
        10. [H] zip funktsioon ([1;3] [2;4] = [(1,2);(3,4)]) (projekt);
        11. [H] elemendi indeks järjendis (projekt);
    - Ülesannete klassifikatsioon tehniliste oskuste põhjal
        - lineaarne- ja sabarekursioon:
            1. [HL] kas sisse *map* on sabarekursiivne?;
            2. [HL] kas sisse *fold* on sabarekursiivne?;
            3. [HL] kas funktsioon *fold_right* on sabarekursiivne?;
            4. [HL] koodi põhjal sabarekursiivsuse tuvastamine (map);
            5. [H] Ackermanni funktsioon;
        - baasjuhu ja sammu äratundmine ning mõistmine:

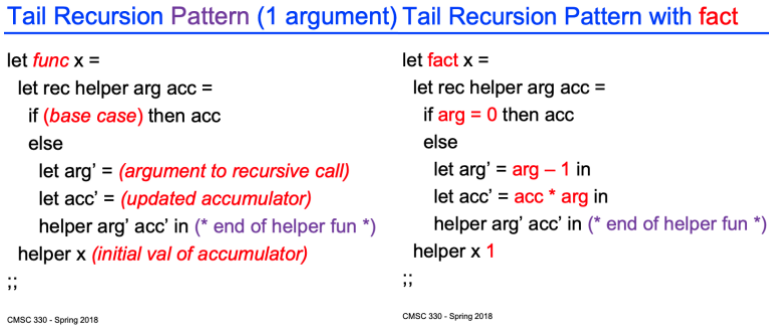
1. [H] Eukleidese algoritm – suurim ühistegur;
2. [H] Ackermanni funktsioon;
3. [H] järjendi elementide summa kuni argumendina antud indeksini;
4. [H] järjendi elementide summa kuni argumendina antud indeksijärjendini;
5. [H] zip funktsioon;
6. [H] elemendi indeks järjendis;

- alamülesandest terve ülesande lahenduseni jõudmine:
  1. [H] Eukleidese algoritm – suurim ühistegur;
  2. [H] Ackermanni funktsioon;
  3. [H] järjendi elementide summa kuni argumendina antud indeksini;
  4. [H] järjendi elementide summa kuni argumendina antud indeksijärjendini;
  5. [H] zip funktsioon;
  6. [H] elemendi indeks järjendis;
- rekursiooni vähenemine baasjuhu suunas:
  1. [H] Eukleidese algoritm – suurim ühistegur;
  2. [H] Ackermanni funktsioon;
  3. [H] järjendi elementide summa kuni argumendina antud indeksini;
  4. [H] järjendi elementide summa kuni argumendina antud indeksijärjendini;
  5. [H] zip funktsioon;
  6. [H] elemendi indeks järjendis;
- funktsiooni käitumine ootamatute argumentide korral:
  1. [H] zip funktsioon (kui üks järjenditest on tühi, siis ei tee enam paare);
  2. [H] elemendi indeks järjendis (kui elementi ei olegi);
- koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine:
  1. [H] järjendi elementide summa kuni argumendina antud indeksijärjendini (eelmise ülesande väike täiendus);
- algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
  1. [H] Eukleidese algoritm – suurim ühistegur;
  2. [H] järjendi elementide summa kuni argumendina antud indeksini;
- rekursiooni ajaline keerukus;
  1. [HL] sabarekursiivse *map*i ajaline keerukus;

- Tähelepanekuid ja häid ideid
  - Miks on sabarekursioon oluline?
    1. “Pushing a call frame for each recursive call when operating on a järjend is dangerous
       1. – One stack frame for each järjend element
       2. – Big järjend = stack overflow!” (11. loeng)
    2. “So: favor tail recursion when inputs could be large (i.e., recursion could be deep). E.g., – Prefer List.fold_left to List.fold_right • Library

documentation should indicate tail recursion, or not – Convert recursive functions to be tail recursive" (11. loeng)

- Üheargumendiga sabarekursiivse funktsiooni mall (ja näide faktoriaaliga [slaididel veel näide järjendi ümber pooramisega]) (11. loeng):



- Mõiste continuation-passing style ("convert an arbitrary program into an equivalent one, except where no call ever returns") (11. loeng).
- Projektide ja praktikumide juures oli mainitud selgelt ära, kus peaks rekursiooni kasutama ja kus mitte.

CMSC 351 – Algorithms (4. semester)

Õpiväljundid ja aine sisu http://www.cs.umd.edu/~samir/251/251.html:

- "Each student is expected to know the basic concepts of programming (e.g. loops, pointers, recursion"
- Rekursioon eraldi ei mainita.

- Õppeviisid: loengud
- Õppematerjalid:
  - Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2009). *Introduction to Algorithms* (3rd ed.). MIT Press (Any edition is fine)
  - Kursuseleht: http://www.cs.umd.edu/class/spring2018/cmsc351/
  - Loenguslaidid: https://www.cs.umd.edu/users/meesh/cmsc351/mount/lectures/
- Tähelepanekuid ja häid ideid
  - Rekursiooni definitsiooni ei korrata loenguslaididel üle.
  - Rekursiooni mainitakse.
  - AA sarnane kursus.

California Ülikool, Berkeley

Kõik ained: http://guide.berkeley.edu/courses/compsci/
Kõik kursuselehed: http://inst.eecs.berkeley.edu/classes-eecs.html

CS 61 A The Structure and Interpretation of Computer Programs (1. semester)

- Õppeviisid https://cs61a.org/articles/about.html:
  - 3*50 minutit loeng iga nädal, praktikumid, arutelud, grupimentordamine, üks-ühele tuutordamine;
- Õppematerjalid:

- Kursuse koduleht: https://cs61a.org/
- e-raamatu peatükk „Recursive Functions“: http://composingprograms.com/pages/17-recursive-functions.html
- e-raamatu peatükk “Recursive Objects”: http://composingprograms.com/pages/29-recursive-objects.html
- 9. loeng: https://drive.google.com/drive/u/0/folders/1xXLVM4tpausgorkVyWjf8_fwAQN24fHw;
- 10. loeng: https://docs.google.com/presentation/d/1UnAq6kfCV1QmcN1_xGxQdJl7Nlpe-sx1YbACzLDuGL0/edit#slide=id.p5;
- 3. diskussioon: https://cs61a.org/disc/disc03.pdf;
- 3. kodutöö: https://cs61a.org/hw/hw03/
- loendamisprobleemide tööleht: https://sequoia-tree.github.io/TEMPORARY/Counting%20Problems.pdf;
- varasemad eksamiküsimused: https://cs61a.org/resources.html

- Esitatud ülesanded
  - Kõik ülesanded:
    1. [HL] naturaalarvu paarsuskontroll;
    2. [H] kahe numbri korrutis (3. diskussioon);
    3. [H] astendamine (n, m) (3. diskussioon);
    4. [H] kas on algarv? (3. diskussioon);
    5. [H] arv erinevaid viise minna üles n-astmelisest trepist, kui korraga saab teha 1 või 2 sammu (3. diskussioon);
    6. [H] arv erinevaid viise minna üles n-astmelisest trepist, kui korraga saab teha 1 kuni k (k.a.) sammu (3. diskussioon);
    7. [H] arv sisaldab arvu ‘7’ (3. kodutöö);
    8. [H] pingpongi jada (3. kodutöö);
    9. [H] arv mitu erinevat viisi saab tulevikumünte (1 ja kahe astmed) kindla summa puhul jaotada (3. kodutöö);
    10. [HV] Hanoi tornid (3. kodutöö);
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal
    - iteratiivselt lahendatud probleemi rekursiooniga lahendamine:
      1. [H] kas arv on algarv?;
      2. [H] pingpongi jada (kui kohe rekursiooniga hakkama ei saa);
    - puurekursioon (sh binaarne rekursioon):
      1. [HL] arv erinevaid viise minna üles n-astmelisest trepist, kui korraga saab teha 1 või 2 sammu;
      2. [HL] arv erinevaid viise minna üles n-astmelisest trepist, kui korraga saab teha 1 kuni k (k.a.) sammu;
    - mitmepoolne rekursioon:
      1. [HL] naturaalarvu paarsuskontroll;
    - baasjuhu ja sammu äratundmine ning mõistmine:
      1. [HL] ristsumma;

2. [H] kahe numbri korrutis;
3. [H] kas on algarv?;
4. [HL] arv erinevaid viise minna üles n-astmelisest trepist, kui korraga saab teha 1 või 2 sammu;
5. [HL] arv erinevaid viise minna üles n-astmelisest trepist, kui korraga saab teha 1 kuni k (k.a.) sammu;
6. [H] arv sisaldab arvu '7';
7. [H] pingpongi jada;
8. [H] arv mitu erinevat viisi saab tulevikumünte (1 ja kahe astmed) kindla summa puhul jaotada;
9. [HV] Hanoi tornid;

- alamülesandest terve ülesande lahenduseni jõudmine:
  1. [HL] ristsumma;
  2. [H] arv sisaldab arvu '7';
  3. [H] pingpongi jada;
  4. [H] arv mitu erinevat viisi saab tulevikumünte (1 ja kahe astmed) kindla summa puhul jaotada;
  5. [HV] Hanoi tornid;
- rekursiooni vähenemine baasjuhu suunas:
  1. [H] kahe numbri korrutis;
  2. [H] arv erinevaid viise minna üles n-astmelisest trepist, kui korraga saab teha 1 või 2 sammu;
  3. [H] arv sisaldab arvu '7';
  4. [H] pingpongi jada;
  5. [H] arv mitu erinevat viisi saab tulevikumünte (1 ja kahe astmed) kindla summa puhul jaotada;
  6. [HV] Hanoi tornid;
- rekursiivne funktsioonikutse programmeerimiskeskkonna seisukohast:
  1. [H] astendamine (n, m);
- koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine:
  1. [HL] arv erinevaid viise minna üles n-astmelisest trepist, kui korraga saab teha 1 kuni k (k.a.) sammu (sarnane 1 või 2 sammu korraga ülesandele);
- algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
  1. [HL] naturaalarvu paarsuskontroll;

- Tähelepanekuid ja häid ideid
  - Rekursioon neljandal nädalal st. kohe peale iteratsiooni.
  - Miks rekursiooni võib tegelikult olla lihtsam kirja panna (e-raamat):
    1. "Recursive functions leverage the rules of evaluating call expressions to bind names to values, often avoiding the nuisance of correctly assigning local names during iteration. For this reason, recursive functions can be easier to define correctly. However, learning to recognize the computational processes evolved by recursive functions certainly requires practice."

- Mitmepoolne rekursioon -> üheks rekursiivseks funktsiooniks (e-raamat):
    1. “Mutually recursive functions can be turned into a single recursive function by breaking the abstraction boundary between the two functions. In this example, the body of is_odd can be incorporated into that of is_even, making sure to replace n with n-1 in the body of is_odd to reflect the argument passed into it:”
- Millal on funktsioon puurekursioon (sh binaarne rekursioon) ja millal seda kasutada? (3. diskussioon):
    1. “A function with multiple recursive calls is said to be *tree recursive* because each call branches into multiple smaller calls, each of which branches into yet smaller calls, just as the branches of a tree become smaller but more numerous as they extend from the trunk.” (e-raamat);
    2. “As a general rule of thumb, whenever you need to try multiple possibilities at the same time, you should consider using tree recursion.” (3. diskussioon);
- “When learning to write recursive functions, put the base cases first” (10. loeng)
- Kolm üldist sammu rekursiooni definitsioonis (3. diskussioon):
    1. “There are three common steps in a recursive definition:
        1. Figure out your base case: The base case is usually the simplest input possible to the function. For example, factorial(0) is 1 by definition. You can also think of a base case as a stopping condition for the recursion. If you can’t figure this out right away, move on to the recursive case and try to figure out the point at which we can’t reduce the problem any further.
        2. Make a recursive call with a simpler argument: Simplify your problem, and assume that a recursive call for this new problem will simply work. This is called the “leap of faith”. For factorial, we reduce the problem by calling factorial(n-1).
        3. Use your recursive call to solve the full problem: Remember that we are assuming the recursive call works. With the result of the recursive call, how can you solve the original problem you were asked? For factorial, we just multiply $(n - 1)!$ by n.”
- Rekursiivsed objektid: Linked List Class, Tree Class, Sets (http://composingprograms.com/pages/29-recursive-objects.html)
- Loendamisprobleemid. Kuidas läheneda loendamisprobleemidele? (loendamisprobleemide tööleht)
    1. “Sometimes it's necessary to make multiple recursive calls, in order to solve one problem. Counting problems are a good example of this fact, and in this chapter we will learn how to solve them. First we'll talk about how to plan your base case, and then we'll move on to addressing the recursive calls.”
    2. “The same approach applies to any problem where you're counting some quantity — whether that's how many paths caterpie can take across a grid,

or how many ways you can add coins to make a dollar. Here's the general formula:"

3. "Base Case
    1. Find each parameter's individual base case. This is the smallest value that could be passed in for each parameter.
    2. Make a chart that maps out whether or not each parameter is in its base case.
    3. Fill in all the boxes where at least one parameter is in its base case. Be careful with this one; sometimes it can be tricky. For example, think of count_paths(1, 1). Many students are inclined to say there are zero ways for caterpie to get to the top right corner, when in fact there is one way: for caterpie to not move at all.
4. Recursive Call
    1. Look at the problem definition. It will explain how you're allowed to break the problem into smaller ones. Typically you can make a problem smaller by making some sort of decision. Does caterpie go up, or right? Do you use a dime, or something else?
    2. For each smaller problem, make a recursive call with parameters that correspond to what the decision was.
    3. Combine those recursive calls somehow, depending on what you want to do. For example you might add them up, multiply them, or put them in a list."

CS 61 B Data Structures (2. semester)

- Õpiväljundid ja aine sisu (e-raamat):
    - Õpiväljundites rekursiooni välja ei tooda.
    - "This course presumes that you already have a strong understanding of programming fundamentals. At the very least, you should be comfortable with the ideas of object oriented programming, recursion, lists, and trees."
- Õppeviisid:
    - Loengud, iganädalased arutulussessioonid ja praktikumid, vitamiinid (iganädalased lühiküsimused eelneva nädala materjalide peale).
    - Youtube videod toetamaks õppimist (https://joshhug.gitbooks.io/hug61b/content/chap2/chap21.html)
- Õppematerjalid:
    - Kursuse veebileht: https://sp18.datastructur.es/
    - e-raamat: https://joshhug.gitbooks.io/hug61b/content/
    - "All textbooks for this course are optional."
        1. Paul Hilfinger's (free) Java Reference
        2. Head First Java, 2nd Edition by Sierra and Bates (O'Reilly, 2005).
    - 3. loeng: https://docs.google.com/presentation/d/1c1AQqfCDh2znyIkJM45N2zUq9ww_NN-2Gi9JQkyRROs/edit#slide=id.g825e60afb_0_76
    - 3. loengu märkmed: https://sp18.datastructur.es/materials/lectures/lec3/lec3.html

- Esitatud ülesanded
  - Kõik ülesanded:
    1. [H] järjendi elementide arv;
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal
    - baasjuhu ja sammu äratundmine ning mõistmine:
      1. [H] järjendi elementide arv;
- Tähelepanekuid ja häid ideid
  - 3. loengul küsitakse tudengitelt loenguslaididel, kuidas nad enda taste rekursiooniga hindavad (*very comfortable; comfortable; somewhat comfortable; I have never done this*);
  - Selection Sort: https://joshhug.gitbooks.io/hug61b/chap3/chap31.html
  - Self-paced materjalid, millele viidatakse:
    1. Pythoni juures (hea materjal veel rekursiooni kohta): https://selfpaced.bitbucket.io/#/python/calendar

CS 61 BL Data Structures and Programming Methodology (2. semester)

- Õpiväljundid ja aine sisu:
  - "CS 61A is an important prerequisite. We expect to build heavily on data-oriented and object-oriented design approaches introduced in this course and on algorithms for recursive list and tree manipulation. … We assume you are coming in with zero Java experience, but we will move through basic Java syntax very quickly."
- Õppeviisid:
  - Loengud, praktikumid, test iga praktikumi alguses
- Õppematerjalid:
  - Kursuse veebileht: https://cs61bl.org/su18/
  - Kursuse märkmed: https://joshhug.gitbooks.io/hug61b/
  - Kursuse märkmed 31. ptk: https://joshhug.gitbooks.io/hug61b/chap3/chap31.html
  - Kursuse märked 21. ptk: https://joshhug.gitbooks.io/hug61b/chap2/chap21.html
  - Kohustuslikku lugemist ei ole. Küll aga peab iga praktikumi sisu lugema.
- Esitatud ülesanded
  - Kõik ülesanded:
    1. [H] järjendi elementide arv (kursuse märked 21. ptk);
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal
    - baasjuhu ja sammu äratundmine ning mõistmine:
      1. [H] järjendi elementide arv;
    - alamülesandest terve ülesande lahenduseni jõudmine:
      1. [H] järjendi elementide arv;
    - rekursiooni vähenemine baasjuhu suunas:
      1. [H] järjendi elementide arv;
- Tähelepanekuid ja häid ideid
  - Rekursiooni definitsiooni/ olemust ei korrata üle.
  - Rekursioon andmestruktuuril, mis ei ole loomupäraselt rekursiivne (kursuse märkmed 31. ptk):

        - Recursive Helper Methods: “This approach is quite common when trying to use recursion on a data structure that is not inherently recursive, e.g. arrays”

CS61AS self-paced (ei ole kohustuslik kursus)
https://berkeley-cs61as.github.io/syllabus.html
e-õpik: https://berkeley-cs61as.github.io/textbook/common-recursive-patterns.html

- Kolm harilikku rekursioonimustrit, mis võivad aidata probleemi lahendada (e-õpik):
    - The "Every" Pattern (“collect the results of transforming each element of a word or sentence into something else.”
        - most “every” patterns have two cases
        - we only have to distinguish between the base case and the recursive case
    - The "Keep" Pattern (“choosing some of the elements and filtering out the others.”)
        - “the "keep" procedures have three cases”
        - “there is still a base case, but there are two recursive cases: we have to decide whether or not to keep the first available element in the return value. When we do keep an element, we keep the element itself, not some function of the element.”
    - The "Accumulate" Pattern (“combines all of the elements of the argument into a single result”)
        - “We're using some combiner (+ or word) to connect the word we're up to with the result of the recursive call”
        - “The base case tests for an empty argument, but the base case return value must be the identity element of the combiner function.”

Princetoni Ülikool

COS 126 Computer Science: An Interdisciplinary Approach (1. semester)

- Õpiväljundid ja aine sisu:
    - Rekursiooni ei mainita.
- Õppeviisid:
    - Loengud, praktikumid
- Õppematerjalid:
    - R. Sedgewick and K. Wayne, *Computer Science: An Interdisciplinary Approach*, Addison–Wesley Professional, 2016. ISBN 978-0134076423. We will be referencing this text all semester. The lectures are based on its contents.
    - E-raamat (ptk. 23 sisaldab üle 60 harjutuse, millest on arvesse esimesed 30): https://introcs.cs.princeton.edu/java/23recursion/;
    - 6. loeng: http://www.cs.princeton.edu/courses/archive/spring19/cos126/lectures/CS.6.Recursion.2x2.pdf;

- 6. praktikum: https://docs.google.com/presentation/d/1RIdUvwhMRb1pN5O040wkXA3l2QKlgvtKWtxjkrwsPvc/edit#slide=id.g4397ec55f0_0_1;
- kodutööd: https://introcs.cs.princeton.edu/java/assignments/;

- Esitatud ülesanded
  - Kõik ülesanded:
    1. [N, H] täisarv binaararvuks (e-raamat);
    2. [N, H] Hanoi tornid (3 versiooni) (e-raamat);
    3. [N, H] fraktaal: H-puud (e-raamat);
    4. [N, H] Fibonacci (tavaline ja kontroll, kas n on Fibonacci järjendis);
    5. [H] grupiarutlus rekursiooniteemaliste küsimustega;
    6. [H] Sierpinski kolmnurk (kodutöö);
    7. [H] iseenda fraktaali joonistamine;
    8. [H] suurim ühisosa;
    9. [H] müsteerium-funktsioon (x9);
    10. [H] permutatsioonid (sisendparameetriga n, aga ka (n, k));
    11. [H] kombinatsioonid (C(n, k));
    12. [H] fraktaal: rekursiivsed ruudud;
    13. [HV] Collatzi funktsioon (kasuta memoizationit!);
    14. [H] Browi saar;
    15. [H] McCarthy funktsioon;
    16. [H] fraktaal: rekursiivne puu;
    17. [H] Eukleidese algoritm (tavaline ja pi ennustamine);
    18. [H] Golden ratio;
    19. [H] Pelli arvud;
    20. [H] arvu a b-ks muutmine kasutades võimalikult vähe liitmist (arvuga 1) ja korrutamist (arvuga 2) (nt. 5 23 -> 23 = ((5 * 2 + 1) * 2 + 1)));
    21. [H] fraktaal: Hadamardi maatriks;
    22. [H] malelauale 8 kuninganna asetamine nii, et nad üksteisele tuld ei annaks;
    23. [H] keeruline kordus (“F(0) = 0 and F(n) = n - F(F(n-1)). What is F(100000000)?”);
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal
    - rekursiivselt lahendatud probleemi iteratiivselt lahendamine:
      1. [H] Golden ratio;
    - mõttes etteantud funktsiooni väärtustamine olenevalt sisendist:
      1. [H] suurim ühisosa;
      2. [H] müsteerium-funktsioon (x9);
      3. [H] McCarthy funktsioon;
    - baasjuhu ja sammu äratundmine ning mõistmine:
      1. [H] Hanoi tornid (illustreeriv funktsioon);
      2. [H] grupiarutlus rekursiooniteemaliste küsimustega;
      3. [H] Sierpinski kolmnurk;
      4. [H] iseenda fraktaali joonistamine;

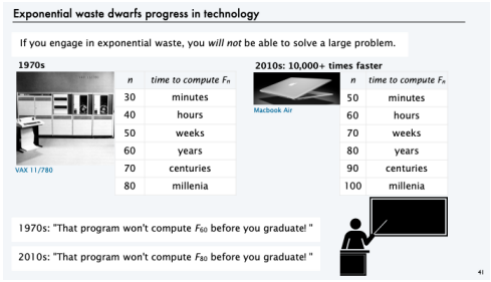
5. [H] permutatsioonid;
6. [H] kombinatsioonid (C(n, k));
7. [H] Pelli arvud;
8. [H] arvu a b-ks muutmine;
9. [H] malelauale 8 kuninganna asetamine nii, et nad üksteisele tuld ei annaks;

- alamülesandest terve ülesande lahenduseni jõudmine:
  1. [H] Sierpinski kolmnurk (kodutöö);
  2. [H] iseenda fraktaali joonistamine;
  3. [H] permutatsioonid;
  4. [H] kombinatsioonid (C(n, k));
  5. [H] fraktaal: rekursiivsed ruudud;
- rekursiooni vähenemine baasjuhu suunas:
  1. [H] grupiarutlus rekursiooniteemaliste küsimustega (Collatzi puhul ei tea seda!);
  2. [H] Sierpinski kolmnurk (kodutöö);
  3. [H] iseenda fraktaali joonistamine;
  4. [H] permutatsioonid;
  5. [H] kombinatsioonid (C(n, k));
  6. [H] fraktaal: rekursiivne puu;
- funktsiooni käitumine ootamatute argumentide korral:
  1. [H] Eukleidese algoritm (negatiivsed sisendid);
- koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine:
  1. [H] fraktaal: H-puud;
  2. [H] täisarv binaararvuks;
  3. [H] permutatsioonid (sisendparameetriga (n, k));
  4. [H] Browi saar;
  5. [H] müsteerium-funktsioon (mis muutub?));
- algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
  1. [H] Sierpinski kolmnurk (kodutöö);
  2. [H] fraktaal: H-puud;
  3. [H] fraktaal: rekursiivsed ruudud;
  4. [H] fraktaal: rekursiivne puu;
  5. [H] Golden ratio;
  6. [H] Fibonacci (mitteklassikalise definitsiooni järgi);
  7. [H] Pelli arvud;
  8. [H] fraktaal: Hadamardi maatriks;
  9. [H] malelauale 8 kuninganna asetamine nii, et nad üksteisele tuld ei annaks;
  10. [H] keeruline kordus;
- rekursiooni ajaline keerukus;
  1. [N] Hanoi tornid (“Note: Recursive solution has been proven optimal”);
  2. [HV] Collatzi funktsioon;

3. [H] Fibonacci (suurim n < 1min, tavaline vs mitteklassikaline);

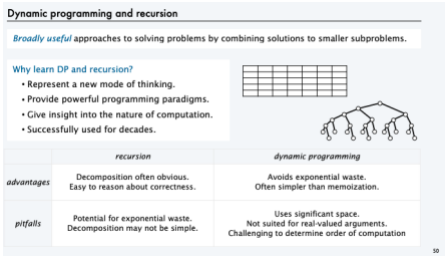
- Tähelepanekuid ja häid ideid
  - Ülesannetes: üks näide ja siis hea posu harjutusi.
  - Miks rekursiooni õppida? (6. loeng)
    1. "Represents a new mode of thinking.
    2. Provides a powerful programming paradigm.
    3. Enables reasoning about correctness.
    4. Gives insight into the nature of computation."
  - Päriselu näiteid (6. loeng): "Many computational artifacts are naturally self-referential."
    1. File system with folders containing folders.
    2. Fractal graphical patterns.
    3. Divide-and-conquer algorithms (stay tuned).
  - Kuidas same rekursiooni funktsioonis kindlad olla (6. loeng)?
    1. Kasutagem matemaatilist induktsiooni.
    2. Kiire tagasivaade sellele:
       1. "To prove a statement involving a positive integer N • Base case. Prove it for some specific values of N. • Induction step. Assuming that the statement is true for all positive integers less than N, use that fact to prove it for N." (6. loeng)
       2. Näide: esimese n paaritu täisarvu summa on n ruudus.
    3. Tõestuse näide slaididel.

  - Tüüpilised vead (6. loeng):
    1. "Missing base case
    2. No convergence (koondumine) guarantee
    3. Both lead to infinite recursive loops"
  - Päriselunäiteid:
    1. murdosaline Browni liikumine
       1. A process that models many phenomenon.
       2. Price of stocks.
       3. Dispersion of fluids.
       4. Rugged shapes of mountains and clouds.
       5. Shape of nerve membranes.
    2. Fibonacci ("Models many natural phenomena and is widely found in art and architecture):
       1. golden ratio;
       2. "Model for reproducing rabbits.
       3. Nautilus shell.
       4. Mona Lisa"
- ***exponential waste*** – miks see oluline on ja kuidas seda vältida:



- Kuidas vältida: “Memoization
    1. Maintain an array memo[] to remember all computed values.
    2. If value known, just return it.
    3. Otherwise, compute it, remember it, and then return it.”

- Dünaamiline programmeerimine:
    - “An alternative to recursion that avoids recomputation”
    - “Build computation from the "bottom up".
    - Solve small subproblems and save solutions.
    - Use those solutions to build bigger solutions.”
    - “Key advantage over recursive solution. Each subproblem is addressed only once”



- [H] Grupiarutlus rekursiooniteemaliste küsimustega (6. praktikum):
    - “What does it mean to say a program uses recursion?
    - What are the **two components** of a recursion?
    - How is the **Collatz** algorithm different from most other recursive algorithms?”
- “Discover a connection between the golden ratio and Fibonacci numbers. *Hint*: consider the ratio of successive Fibonacci numbers: 2/1, 3/2, 8/5, 13/8, 21/13, 34/21, 55/34, 89/55, 144/89, ...”
- “Prove by mathematical induction that the alternate definitions of the Fibonacci function given in the previous two exercises are equivalent to the original definition.”

COS 226 Algorithms and Data Structures: (3. semester)

- Õpiväljundid ja aine sisu:
    - Rekursiooni ei mainita.
    - “COS 126” on eeldusaine.
- Õppeviisid:
    - loengud, laboratooriumid, kodutööd, testid, eksam, iClickerid
- Õppematerjalid:
    - *Algorithms, 4th edition* by Robert Sedgewick and Kevin Wayne. Addison-Wesley Professional, 2011, ISBN 0-321-57351-X. The assigned readings are required.

        - Kursuse koduleht: https://www.cs.princeton.edu/courses/archive/fall18/cos226/syllabus.php
        - Loenguslaidid: https://www.cs.princeton.edu/courses/archive/fall18/cos226/lectures.php
        - Loenguvideod
        - Arutlusfoorum
    - Tähelepanekuid ja häid ideid
        - Rekursiooni olemusest ei käida aines üle.

COS 326 Functional Programming (4. semester)

- Õpiväljundid ja aine sisu:
    - Rekursiooni ei mainita.
- Õppeviisid:
    - loengud, ettekirjutused, kodutööd
    - Keel: OCaml
- Õppematerjalid:
    - Kursuse koduleht: https://www.cs.princeton.edu/courses/archive/fall18/cos326/
    - 6. probleemihulk: https://www.cs.princeton.edu/courses/archive/fall18/cos326/ass/a6.php
    - E-õpik: https://www.cs.princeton.edu/courses/archive/fall18/cos326/notes/recursion.php
- Esitatud ülesanded
    - Kõik ülesanded:
        1. [HV] mäluga Fibonacci (6. probleemihulk);
    - Ülesannete klassifikatsioon tehniliste oskuste põhjal
        - mitmepoolne rekursioon:
            1. [HV] mäluga Fibonacci;
        - baasjuhu ja sammu äratundmine ning mõistmine:
            1. [HV] mäluga Fibonacci;
        - koodist arusaam st. olemasoleva rekursiivse funktsiooni täiendamine:
            1. [HV] mäluga Fibonacci;
- Tähelepanekuid ja häid ideid
    - Millal rekursiooni tarvis on (e-õpik):
        - “Data structures such as lists, trees and graphs may be arbitrarily large. They are recursively defined data structures, and we do need recursive functions to get at all the information they contain.”
        - “Recursive functions often arise when one must process recursive data. Both natural numbers and lists may be viewed as recursive data. Indeed, they are structurally very similar types:
        - Natural Numbers:
            1. 0 is a natural number
            2. 1+m is a natural number when m is a natural number
        - Lists:
            1. [] is a list

2. hd::tail is a list when tail is a list.
   - The similarity in the structure of the values of each type leads to (somewhat) structurally similar programs."

MIT

Õppekava: http://www.eecs.mit.edu/curriculum2017

6.0001 Introdution to Computer Science and Programming in Python (määratlemata/ 1. semester)

- Õpiväljundid ja aine sisu:
  - Rekursiooni ei mainita, aga on kalendris kirjas (6/12).
  - Ei eelda mingit programmeerimiskogemust.
- Õppeviisid:
  - loengud, interaktiivsed koduülesanded.
  - in-class küsimusi ei olnud.
- Õppematerjalid:
  - Guttag, John. *Introduction to Computation and Programming Using Python: With Application to Understanding Data Second Edition*. MIT Press, 2016. ISBN: 9780262529624.
  - Kursuse koduleht: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/index.htm
  - 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
  - 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/
- Esitatud ülesanded
  - Kõik ülesanded:
    1. [HV] sõne permutatsioonid (4. probleemihulk);
    2. [H] Caesari šiffer 4. probleemihulk;
  - Ülesannete klassifikatsioon tehniliste oskuste põhjal
    - baasjuhu ja sammu äratundmine ning mõistmine:
      1. [HV] sõne permutatsioonid;
    - rekursiooni vähenemine baasjuhu suunas:
      1. [HV] sõne permutatsioonid;
    - rekursiivse ja iteratiivse lahendusmeetodi vahel eelistuse tegemine:
      1. [H] Caesari šiffer;
- Tähelepanekuid ja häid ideid
  - Huvitav kahene definitsioon (6. loeng): "What is recursion?

1. Algorithmically: a way to design soluSons to problems by divide-and-conquer or decrease-and-conquer ◦ reduce a problem to simpler versions of the same problem
2. Semantically: a programming technique where a func0on calls itself
    1. in programming, goal is to NOT have infinite recursion
    2. must have 1 or more base cases that are easy to solve
    3. must solve the same problem on some other input with the goal of simplifying the larger problem input"

- Tõestamine (6. loeng): "Mathematical induction
    1. To prove a statement indexed on integers is true for all values of n:
        1. Prove it is true when n is smallest value (e.g. n = 0 or n = 1)
        2. Then prove that if it is true for an arbitrary value of n, one can show that it must be true for n+1"
- Palindroom (6. loeng). "Recursion on nonnumerics
    1. First, convert the string to just characters, by stripping out punctuation, and converting upper case to lower case
    2. Then base case: a string of length 0 or 1 is a palindrome
    3. Recursive case:
        1. If first character matches last character, then is a palindrome if middle section is a palindrome"
- Permutatsioonide ülesande soovitatud lahenduskäik (4. probleemihulk):
    1. "In order to solve any recursive problem, we must have at least one base case and a recursive case (or cases). We can think of our base case as the simplest input we could have to this problem (for which determining the solution is trivial and requires no recursion) -- for this approach, our base case is if sequence is a single character (there's only one way to order a single character)."
    2. Pseudokood.
- Iga ülesande viimaseks osaks on testide kirjutamine (4. probleemihulk).

6.006 Introduction to Algorithms (määramata/ 3. semester)

- Õpiväljundid ja aine sisu https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/syllabus/:
    - Rekursiooni ei mainita.
    - Eelduseks: "A firm grasp of Python and a solid background in discrete mathematics are necessary prerequisites to this course. "
- Õppeviisid:
    - loengud, retsitatsioonid, probleemihulgad, kaks testi, koodiülesanded, kirjalikud ülesanded
- Õppematerjalid:
    - Cormen, Thomas, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms*. 3rd ed. MIT Press, 2009. ISBN: 9780262033848.

    - Kursuse koduleht: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/index.htm
    - Probleemihulgad:
        1. 1. probleemihulk: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/assignments/MIT6_006F11_ps1.pdf;
        2. 2. probleemihulk: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/assignments/MIT6_006F11_ps2.pdf.
- Esitatud ülesanded
    - Kõik ülesanded:
        1. [H] relatsiooni asümptootilise käitusaja määramine (1. probleemihulk);
        2. [H] fraktaal: Kochi lumehelves (2. probleemihulk);
    - Ülesannete klassifikatsioon tehniliste oskuste põhjal
        - algoritmi ja/või matemaatilise definitsiooni põhjal funktsiooni kirjapanek:
            1. [H] fraktaal: Kochi lumehelves;
        - rekursiooni ajaline keerukus;
            1. [H] relatsiooni asümptootilise käitusaja määramine (x4);
            2. [H] fraktaal: Kochi lumehelves (x3);
        - aktiveerimiskirjete ja/või rekursioonipuu joonistamine funktsiooni väljakutse põhjal:
            1. [H] fraktaal: Kochi lumehelves (puu kõrgus) (x4);
            2. [H] fraktaal: Kochi lumehelves (puu tippude arv tasemel i) (x4);
- Tähelepanekuid ja häid ideid
    - Rekursiooni olemust uuesti üle ei käida.
    - Ülesanded on tehnilisemad (nt. võetakse arvesse CPU kasutust, erinevate programmide mälukasutust, renderdamist) ja nõuavad tihti viimase osana tõestust.