TARTU ÜLIKOOL
ARVUTITEADUSE INSTITUUT

Programmeerimisharjutuste lähteülesannete kogu

Versioon 4.4 19. märts 2021. a. 11:04

Koostajad: Ahti Põder
Jüri Kiho

Tartu 2021

Käesoleva õppevahendi väljaandmist on toetanud . . .

Toimetamine: Jüri Kiho

ISBN . . .

Sisukord

Saateks

Käesolev materjal kujutab endast ülesannete kogu eeskätt ülikoolikursuse *Programmeerimisharjutused* tarbeks.

Ülesanded on reeglina sõnastatud programmeerimiskeeltest ja neis kasutatavatest andmestruktuuridest sõltumatult. Tegemist on nimelt lähteülesannetega, millest lähtuvalt saab (juhendaja) sõnastada mitmesuguseid konkreetseid programmeerimisülesandeid.

Näiteks, lähteülesandest

- Leida antud järjendi suurima summaga alamjärjend.

saadud paar programmeerimisülesannet:

- Koostada Python-funktsioon, mis antud arvujärjendi korral tagastab selle (ühe) suurima summaga alamjärjendi algus- ja lõpuindeksi.
- Koostada Java-meetod, mis antud arvujärjendi korral leiab selle suurima summaga alamjärjendi nii jõumeetodil (kõiki alamjärjendeid läbi vaadates) kui ka Kadane algoritmi kohaselt ning tagastab arvupaarina vastavad kaks lahendusaega.

Lähteülesandele, milles nõutakse "Loetleda elemendid, mis ...", võib programmeerimisülesannetes vastata nt "Printida elemendid, mis ... , ja nende arv." või "Tagastada nende elementide järjend, mis ..." vmt.

Rida ülesandeid, mille tekstid osaliselt kattuvad, koondatakse enamasti nn kobarülesandeks kujul

```
<Ühine algustekst>
   (a) <esimese ülesande eritekst>
   (b) <teise ülesande eritekst>
   ...
<ühine lõputekst> (kui on).
```

Näiteks, kaks lähteülesannet

```
Antud kolme erineva arvu korral leida neist suuruselt keskmine.
```

ja

```
Antud kolme erineva arvu korral leida suurima ja vähima erinevus.
```

saab esitada kobarülesandena (ilma ühise lõputekstita)

```
Antud kolme erineva arvu korral leida
   (a) neist suuruselt keskmine;
   (b) suurima ja vähima erinevus.
```

Rõhutame, et kobarülesandega esitatakse niipalju lähteülesandeid, kuipalju loendis on eritekste; kobarülesande eritekstide loendis antakse (tähelised) järjekorranumbrid ümarsulgudes: (a), (b), (c) jne.

Täpsustus: ülesannete tekstides peetakse naturaalarvude all silmas positiivseid täisarve, st vähim naturaalarv on 1, mitte 0.

Käesolevas esitatud läheülesannete allikana on muuhulgas kasutatud programmeerimisülesandeid ainekursuse *Programmeerimisharjutused* 2019 ja 2020 aastate praktikumide ning kontrolltööde materjalidest. Viimastes leiduvate mitmete programmeerimisülesannete sõnastajaks on Härmel Nestra.

Üldiselt, käesolevas esitatud lähteülesanneteks on enamasti juba klassikaliseks kujunenud selle ("Progammeerimisharjutuste") taseme ülesanded, või siis nende pisemad modifikatsioonid. Lähteülesanded on grupeeritud kuude, töötluse suunitluselt lähedastesse jaotistesse. Viimases, lisajaotises esitatakse rekursiivsete programmide koostamise metoodiline juhend konkreetsete, Java-põhiste programmeerimisülesannete rekursiivsete näidislahenduste baasil.

Kogumikku ei ole võetud

– pigem "Algoritmide ja andmestruktuuride" kursusesse kuuluvaid ülesandeid (sh Algoritmide ja andmestruktuuride ülesannete kogust),

– dünaamilisele kavandamisele (dünaamilisele programmeerimisele) orienteeritud ülesandeid,

– samuti paljudest spetsiifilistest, informaatika kõrvalvaldkondadest (nt arvutusmeetodid, matemaatiline statistika, geomeetria, füüsika jt) pärinevaid keerukamaid ülesandeid,

– aga ka kahendpuude erirakendustega (nt kahendotsimispuud, kuhjad) seotud ülesanded.

Ebakorrektsustest ja soovitustest palume teada anda aadressil
`Ahti.Põder@ut.ee`.

1. Arvutusülesanded

1.1. Leida ühest suurem täisarv, mille ruut võrdub selle arvu kümnendnumbrite summa (ristsumma) kuubiga.

1.2. Kontrollida, kas antud kahe naturaalarvu kümnendnumbrite hulgad on ühisosata.

Näiteks,
arvude 9124 ja 35876 kümnendnumbrite hulgad on ühisosata;
arvude 91624 ja 358761 kümnendnumbrite hulkade ühisosa on $\{1,6\}$.

1.3. Loetleda kõik kolmekohalised kümnendsüsteemi arvud, mille ruut on palindroom.

Üks sellistest arvudest on näiteks 264, mille ruut on 69696.

1.4. Antud naturaalarvu n korral leida (kui võimalik) vähim $k \geq 0$ nii, et arvu $n \cdot 3^k$ kolmas kümnendnumber on 0.

Näiteks,
$n = 13$ korral $k = 11$ ($3^{11} = 177147$, $n \cdot 177147 = 2302911$):
$n = 156$ korral $k = 2$ ($3^2 = 9$, $n \cdot 9 = 1404$):
$n = 364$ korral $k = 13$ ($3^{13} = 1594323$, $n \cdot 1594323 = 580333572$);
$n = 1908$ korral $k = 0$ ($3^0 = 1$, $n \cdot 1 = 1908$).

1.5. Antud naturaalarvu n korral leida suurim astendaja k, mille puhul 2^k on arvu n jagaja.

1.6. Antud arvukolmiku korral leida

(a) suurim ja vähim;

(b) nende seas suuruselt keskmine;

(c) milline neist asub kõige lähemal nende kolme arvu aritmeetilisele keskmisele;

(d) neist moodustatud mittekahanevalt sorteeritud kolmik.

1.7. Kontrollida, kas isik pikkusega p cm ja kaaluga k kg on ülekaaluline, alakaaluline või normkaalus (st $p - k = 100 \pm 2$).

1.8. Antud on isiku mass (kg) ja pikkus (m). Arvutada selle isiku

1) kehamassi indeks $kmi = \frac{mass}{pikkus^2}$;

2) kaalulisuse hinne kh,

$$kh = \begin{cases} -1, \text{kui } kmi < 20 \\ 0, \text{kui } 20 \leq kmi < 25 \\ 1, \text{kui } 25 \leq kmi < 30 \\ 2, \text{kui } 30 \leq kmi < 35 \\ 3, \text{kui } 35 \leq kmi \end{cases}$$

(Hinnete selgitused: -1 – alakaal, 0 – normis, 1 – kerge ülekaal, 2 – ülekaal, 3 – suur ülekaal.)

1.9. Kontrollida, kas punkt (a,b) asub ringjoonel $x^2+y^2=r^2$, selle ringi sees või väljaspool seda ringi.

1.10. Kontrollida, kas antud arvud võivad olla mingi kolmnurga küljepikkusteks.

1.11. Kontrollida, kas telliskivi (servadega a, b ja c) mahub läbi ruudukujulise ristlõikega avause seinas (ruudu külje pikkus d, avaus risti seinaga).

1.12. Kontrollida, kas telliskivi (servadega a, b ja c) mahub läbi ristkülikukujulise avause seinas (ristküliku mõõtmed p ja q, avaus risti seinaga).

1.13. Kontrollida, kas punkt (x,y) asub võrgus, mille keskpunkt on koordinaatide alguses ning väline raadius on a ja sisemine b.

1.14. Kontrollida, kas punkt (x,y) kuulub punktidega $(1,0)$, $(0,1)$, $(-2,0)$ ja $(0,-1)$ määratud piirkonda koordinaattasandil.

1.15. Kontrollida, kas antud aastanumber vastab liigaastale. (Liigaasta number jagub 4-ga ja kui jagub 100-ga, siis jagub ka 400-ga.)

1.16. Olgu $h_1,h_2,v_1,v_2 \in \{1,\ldots,8\}$. Kontrollida

(a) kas malelaua ruudud (h_1,v_1) ja (h_2,v_2) on sama värvi;

(b) kas ruudus (h_1,v_1) olev ratsu lööb ruutu (h_2,v_2).

1.17. Leida, mitu purki värvi tuleb osta antud põranda (pikkus a, laius b) värvimiseks. Värvi müüakse 1-liitristes purkides ja kiri purgil väidab, et katvus on $8m^2$.

1.18. Laual on kaks risttahukakujulist pappkarpi mõõtmetega a_1, b_1, c_1 ja a_2, b_2, c_2 sentimeetrit. Kontrollida, kas esimene karp mahub täielikult teise karbi sisse, kui papi paksus on $0,2$ cm.

1.19. Ajaloolased hindavad kaitsevarustuse ja hobusega keskaegse rüütli võitlusvõimet võrdseks kolme tavalise sõjamehe omaga. Lahinguväljal kohtuvad kaks vaenuväge, millest ühte kuulub a rüütlit ja b tavalist sõjameest ning teise c rüütlit ja d tavalist sõjameest. Leida, kumb selle lahingu eeldatavasti võidab.

1.20. Antud naturaalarvu n korral

(a) kontrollida, kas n on täisruut;

(b) kontrollida, kas n on täiuslik (st võrdub oma pärisjagajate summaga);

(c) kontrollida, kas n on algarv;

(d) loetleda n pärisjagajad;

(e) leida n pärisjagajate summa.

(Naturaalarvu n *pärisjagajateks* on kõik arvud, millega n jagub, väljaarvatud arv n ise. Näiteks, arvu 28 pärisjagajateks on $1, 2, 4, 7$ ja 14).

1.21. Leida (kui võimalik) lineaarvõrrandisüsteemi

$$\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$$

reaalarvuline lahend, kui on antud kordajad a_1, b_1, a_2, b_2 ja vabaliikmed c_1, c_2.

1.22. Müügimehele anti valida kolme palgamaksmisskeemi vahel:

(a) 105 eurot nädalas puhast palka;

(b) 1.30 tunnis 40 tunni eest nädalas pluss 10% komisjonitasu müügi pealt;

(c) 15% müügi pealt ilma muu palgata.

Lahendada müügimehe ees seisev probleem: lähtudes oletatavast nädalasest müügist, arvutada töötasu kõigil kolmel juhul (et valida nendest võimalustest parim).

1.23. Teatava ainekursuse kestel toimub 3 tunnikontrolli, millest igaühe eest võib üliõpilane saada kuni 10 punkti, üks suur kontrolltöö, mille maksimaalne punktisumma on 100, ja eksam, mida hinnatakse samuti kuni 100 punktiga. Kursuse lõplik hinne punktides kujuneb järgmiselt: 50% eksamist, 25% kontrolltööst ja 25% tunnikontrollidest.

Seejärel määratakse täheline hinne vastavalt üldkehtivale hindamisskeemile: $91 \ldots 100$ punkti annab A, $81 \ldots 90$ annab B, $71 \ldots 80$ annab C, $61 \ldots 70$ annab D, $51 \ldots 60$ annab E, 50 või vähem annab F.

Leida täheline hinne üliõpilasele, kui on teada tema tunnikontrollide, kontrolltöö ja eksami tulemused.

1.24. Mõni aeg tagasi maksis Hansapank (nüüdne Swedbank) arvelduskonto intressi üks kord aastas, hiljem hakkas seda tegema iga kvartali lõpus. Ülesandeks on välja selgitada, kas selline muudatus oli kasulikum pangale või kliendile.

Eeldame, et aasta alguses pannakse kontole teatav rahasumma, ja et aasta kestel sisse- ega väljamakseid ei toimu. Lihtsuse mõttes võib võtta aasta võrdseks panga-aastaga, milles on 360 päeva. Üks pangaaasta koosneb siis neljast 90-päevasest kvartalist. Intressimäär näitab, mitu protsenti kasvab arvel olev raha ühe pangaaasta jooksul. Lühema perioodi puhul on kasv võrdeline perioodi pikkusega: poole pangaaastase perioodi puhul on kasv kaks korda väiksem, veerandi pangaaastase perioodi puhul neli korda jne. Eeldame veel, et intressimäär jäi pärast intressi maksmises tehtud muudatust samaks.

Antud olgu konto seis aasta alguses ja intressimäär. Sellest lähtuvalt tuleb arvutada konto seis aasta lõpus mõlema meetodi korral.

1.25. Eestvedajate erakond kavatseb korraldada kongressi ja vajab ruume. On olemas Ugandi keskus, mille ruumide üürid üheks päevaks on järgmised: A 100 eur, B 149 eur, C 150 eur, D 180 eur ja E 230 eur. Lisaks on võimalik tellida osalejatele einet (küpsised+vesi): luksuseine maksab 20 eur inimese kohta, tavaline 12 eur. Kõigi osavõtjate pealt arvestatakse teenustasu, mis on 12%; teenustasu arvestatakse ainult eine, mitte ruumi üüri eest. Et parandada maksete laekumist, teeb keskus hinnaalandust, kui makse saabub 10 päeva jooksul. Hinnaalandus sõltub arve suurusest. Kui arve on väiksem kui 700 eur, siis on see 3%, kui arve suurus on 700 ja 1400 eur vahel, siis 4%, ja kui arve suurus ületab 1400 eur, siis 5%.

Arvestada kokku arve suurus (koos hinnaalanduseta ja ilma), kui on teada osavõtjate arv, üüritud ruum, eine liik ja päevade arv.

1.26. Eestvedajate erakond soovib kehtestada riigis astmelise tulumaksu. Vaatleme skeemi, mis lähtub inimese aasta jooksul teenitud tulu kuukaupa keskmisest (tulud eurodes). Tulult vahemikus $0\ldots1000$ on tulumaksumäär 0%, vahemikus $1000\ldots2000$ – 15%, vahemikus $2000\ldots5000$ – 25%, vahemikus $5000\ldots10000$ – 33%, vahemikus $10000\ldots20000$ – 40% ja üle 20000 – 50%. Tulumaksu arvestamiseks jagatakse kuutulu ülaltoodud piiride järgi lõikudeks ning igasse lõiku jääva osa pealt võetakse tulumaksu selle lõigu määra järgi. Näiteks kui tulu on 5200, siis täidab see tervenisti kolm esimest vahemikku, kus maksumäärad on 0%, 15% ja 25%, ning 200 ulatuses neljanda vahemiku, kus maksumäär on 33%. Makstav tulumaks on 966 eurot.

Inimese kuutulu järgi arvutada tulumaksu suurus ja protsent, mille see teenitud tulust moodustab.

1.27. Antud on naturaalarv, mille viimane kümnendnumber ei ole 0. Leida arv, mis on saadud antud arvust selle viimase kümnendnumbri esikohale viimise teel. Näiteks, $123456 \Rightarrow 612345$, $12003 \Rightarrow 31200$.

1.28. Kahe antud naturaalarvu n ja m korral

(a) kontrollida, kas n ja m on sõbralikud arvud;

(b) leida nende arvude suurim ühistegur;

(c) leida nende arvude vähim ühiskordne.

(Kaks naturaalarvu on *sõbralikud*, kui nende pärisjagajate summad on võrdsed. Vt ka ülesanne 1.20)

1.29. Loetleda antud lõiku $[a;b]$ kuuluvad kõik sõbralike arvude paarid ($b \leq 30000$).

1.30. Üks tuntumaid probleeme arvuteoorias on *Goldbachi hüpotees*, mille kohaselt iga 2-st suurem paarisarv on esitatav kahe algarvu summana. Näiteks

$$4 = 2+2$$
$$6 = 3+3$$
$$8 = 3+5$$
$$10 = 3+7 = 5+5$$

Leida antud paarisarvu (>2) lahutus kahe algarvu summaks.

1.31. Nimetame n-kohalist naturaalarvu *Armstrongi arvuks*, kui tema kümnendnumbrite n-ndate astmete summa on võrdne arvu endaga. Armstrongi arv on näiteks $153 = 1^3 + 5^3 + 3^3$.

Loetleda kõik kuni neljakohalised Armstrongi arvud.

1.32. Leida võrrandi $4x+3y-9z=5$ kõik täisarvulised lahendid $0 \le x,y,z \le 100$.

Üks lahenditest on näiteks $(2,2,1)$.

1.33. Antud on naturaalarv n. Tuleb leida Pythagorase kolmikuid, mille liikmeteks on arvust n väiksemad naturaalarvud. Täpsemalt: loetleda võrrandi $x^2+y^2=z^2$ sellised naturaalarvulised lahendid kolmikutena (x,y,z), kus $x,y,z<n$ ja $x<y$.

Näiteks, $n=26$ korral on lahendikolmikuteks $(3,4,5)$, $(5,12,13)$, $(6,8,10)$, $(7,24,25)$, $(8,15,17)$, $(9,12,15)$, $(12,16,20)$, $(15,20,25)$.

1.34. Kuupäev antakse arvukolmikuna $(päev, kuu, aasta)$.

(a) Leida mitmes päev aastas on antud kuupäev.

(b) Leida kahe antud kuupäeva vaheline aeg päevades.

(NB! Liigaasta on selline, mille number jagub 4-ga, välja arvatud juhud, mil see jagub 100-ga, kuid ei jagu 400-ga.)

Näiteks (a), kuupäevale 8. oktoober 2020.a. vastab arvukolmik $(8,10,2020)$ ja see on 282. päev aastas 2020.

1.35. Arvutada

(a) paarisarvulise n korral summa $1\cdot 2+3\cdot 4+5\cdot 6+\ldots+(n-1)\cdot n$;

(b) summa $2\cdot 102+6\cdot 98+10\cdot 94+\ldots+46\cdot 58+50\cdot 54$.

1.36. Üks lihtsaim praktikas kasutatav arvutusmeetod on meetod positiivse arvu u ruutjuure arvutamiseks etteantud täpsusega $\varepsilon>0$.

Kõigepealt leiame alglähendi

$$x := \frac{u+1}{2}$$

ning seejärel arvutame järk-järgult uusi lähendeid eeskirja

$$x := \frac{x+\frac{u}{x}}{2}$$

kohaselt. Kui järjekordse leitud lähendi x korral $|x^2-u|<\varepsilon$, siis lõpetame.

Antud positiivse arvu u korral leida $\sqrt{u}$ täpsusega 0.000001.

1.37. *Algarvukaksikuks* nimetame arvude paari $(x,x+2)$, kus nii x kui ka $x+2$ on algarvud.

Antud on naturaalarv n. Leida viis esimest algarvukaksikut jadas $(n,n+1,\ldots)$.

1.38. Üksikul saarel elab erak. Tema maja on kahe uksega, millest kummagi juures asub alguses 3 paari kingi. Kui erak tahab majast väljuda, siis viskab ta loosi, kummast uksest minna ja paneb jalga selle ukse juures asuvad kingad. Tagasi tulles viskab erak jälle loosi, kummast uksest siseneda, ja jätab kingad selle ukse taha.

Teha kindlaks keskmiselt mitme korra järel peab erak väljuma majast paljajalu, kuna loosiga valitud ukse taga kingi ei ole.

1.39. Võrrandi $f(x) = 0$ lahendamiseks kasutatakse sageli nn. lõigu poolitamise meetodit. Kasutaja annab ette lõigu otspunktid, kus lahend asub. Programm jagab lõigu pooleks ja jätab alles selle poole, mille otspunktides on funktsiooni $f(x)$ märgid erinevad. Edasi jagab jälle lõigu pooleks ja valib selle poole, kus märgid erinevad. Arvutamine lõpeb, kui lõigu pikkus muutub väiksemaks kui lubatav viga.

Leida lõigu poolitamise meetodil funktsiooni

$$f(x) = x^4 + 2x^3 - x - 1$$

nullkoht lõigul $[0,1]$ täpsusega 0.00001.

1.40. Loetleda kõik neljakohalised arvud $ABCD$, mis rahuldavad tingimust

$$ABCD = (AB + CD)^2.$$

1.41. Kuulus uurija Rudolf sai teada, et võõramaine spioon on jätnud pealinna ainsasse täisruudus numbriga trammivagunisse vasakult kolmanda istme alla salateate. Kõik pealinna trammid koosnevad kahest vagunist ja on nummerdatud kolmekohaliste arvudega. Seejuures on esimese vaguni number teise omast 100 võrra väiksem ning esimese vaguni number jagub alati 6-ga ja teise oma 7-ga. Mitu trammi on pealinnas?

Leida, millise numbriga vagunile tuleb Rudolfi käsutuses oleva informatsiooni kohaselt pöörata erilist tähelepanu.

1.42. Leida esimene Fibonacci arv, mis on algarv ja suurem antud arvust n.

1.43. Antud naturaalarvu n korral leida (kui võimalik) kaks täisruutu, mille summa on n.

1.44. Leida vähim naturaalarv, mida saab kahel erineval viisil esitada kahe täisruudu summana.

1.45. Leida vähim naturaalarv, mida saab kahel erineval viisil esitada kahe täiskuubi summana.

1.46. Vaatleme ruutvõrrandit

$$ax^2 + bx + c = 0,$$

mille kordajateks on juhu-täisarvud lõigult $[-10;10]$. Leida tõenäosused, et sellel ruutvõrrandil on

1) kaks reaalarvulist lahendit;

2) üks kahekordne lahend;

3) lahendid puuduvad.

Ülesande lahendamiseks võib vaadata läbi kõik kordajate lubatavate väärtuste kombinatsioonid ning arvutada kõigi kolme juhu osakaalud. Need ongi siis otsitavad tõenäosused.

Selline eksperiment annab muu hulgas võrdlemisi hea ettekujutuse, kui suur osa kõigist ruutvõrranditest on üldse lahenduv. Lisaks tasub uurida, kuidas tõenäosused muutuvad, kui rajade $[-10,10]$ asemele võtta midagi muud, näiteks nõuda, et kõik kordajad oleksid positiivsed.

1.47. Pärast maailma loomist elas maa peal kolme liiki elusolendeid - lohed, maod ja saurused. On teada, et iga lohe sööb hommikusöögiks ühe mao, iga madu sööb lõunasöögiks ühe sauruse ja iga saurus sööb õhtusöögiks ühe lohe. Kui mõne looma jaoks toitu ei jätku, siis ta jääb ilma.

Oletame, et pärast maailma loomist oli maa peal k lohet, l madu ja m saurust. Leida, mitmendal päeval toimus viimane söögikord ning mis liiki loomi ja kuipalju jäi pärast seda järele.

1.48. Firma kavatseb korraldada 521 töötajale nädalavahetuseks ühise väljasõidu. Bussifirmadel on pakkuda kolme tüüpi busse: 21-kohalisi, 41-kohalisi ja 43-kohalisi. Seejuures nõutakse esimese bussi eest 240 eurot, teise bussi eest 336 eurot ja kolmanda bussi eest 400 eurot.

Leida, mitu bussi igast tüübist peab firma tellima, et kõik töötajad ära mahuksid ja koguhind oleks seejuures võimalikult väike.

1.49. Tasandil on rõht- ja püstjoonte abil kujutatud ruudustik. Ruudustiku ruudu koordinaatideks (x, y) loetakse selle ruudu reanumber x ja ruudu järjekorranumber oma reas y.

Olgu antud ruudustiku kahe erineva ruudu koordinaadid. Kui need ruudud asuvad ühel joonel, st kas ühel ja samal horisontaalil, vertikaalil või diagonaalil, siis leida nende vahele jäävate ruutude arv (ilma antud ruute sisse arvestamata) vastavalt kas piki horisontaali, vertikaali või diagonaali.

Näiteid:
Ruudud $(4,4)$ ja $(1,1)$ on ühel joonel ja nende vahele jääb 2 ruutu.
Ruudud $(3,10)$ ja $(1,10)$ on ühel joonel ja nende vahele jääb 1 ruut.
Ruudud $(2,5)$ ja $(1,6)$ on ühel joonel ja nende vahele jääb 0 ruutu.
Ruudud $(5,4)$ ja $(5,4)$ langevad kokku (ei ole erinevad).
Ruudud $(5,4)$ ja $(2,3)$ ei ole ühel joonel.
Ruudud $(7,7)$ ja $(2,3)$ ei ole ühel joonel.
Ruudud $(10,8)$ ja $(6,6)$ ei ole ühel joonel.

1.50. Arvutada antud Fibonacci arvu järjekorranumber (indeks) alates nullist.
Näiteks, Fibonacci arvude $0,5$ ja 55 järjekorranumbrid on vastavalt $0,5$ ja 10.

2. Järjendite töötlemine

Järjend on lõplik jada.

Järjendi *a osajärjend* on mittetühi järjend, mis saadakse, valides järjendist a elemente indeksi kasvades.

Järjendi *a alamjärjend* on mittetühi järjend, mis saadakse, valides järjendist a järjestikuseid elemente, st indeksi kasvades ühe võrra. Alamjärjend on osajärjendi erijuht.

Näiteks,
järjendi $(1,2,3,7,4,5,6)$ neli osajärjendit: $(3,7,5)$, $(2,7,5,6)$, $(1,2,3)$, $(3,7,4,5)$. Neist kaks viimast on ühtlasi ka selle järjendi alamjärjenditeks.

Käesolevas jaotises käsitletakse ainult arvujärjendeid.

I Fikseeritud pikkusega ehk staatilised järjendid

2.1. Loetleda antud järjendi elemendid vaheldumisi algusest ja lõpust.
Näiteks,
$(1,2,3,4,5,6) \to (1,6,2,5,3,4)$, $(1,2,4,8,16,32,64) \to (1,64,2,32,4,16,8)$.

2.2. Leida antud järjendi elementide ruutkeskmine.

2.3. Leida kaks suurimat väärtust antud järjendis.

2.4. Antud järjendis a leida suurim selline element, mis asub rangelt 0 ja 1 vahel. Täpsemalt, leida y, $y = \max\{x \mid x \in a, 0 < x < 1\}$, arvestusega, et $max(\emptyset) = 0$.
Näiteks,
$a = (0.4)$, $y = 0.4$;
$a = (-0.3, 1.5)$, $y = 0$;
$a = (1.0, 0.7, 0.3, -0.9, 0.0)$, $y = 0.7$.

2.5. Olgu antud naturaalarvude järjend $a = (a_1, a_2, \ldots a_n)$. Arvutada s,

$$s = \sum_{i=1}^{n} a_i \cdot (1 + a_i \bmod 2)$$

(s on summa a elementidest, kus paarisarvulise väärtusega elemendid on võetud ühekordselt ja paarituarvulise väärtusega elemendid kahekordselt).
Näiteks,
$a = (4)$, $s = 4$;
$a = (3)$, $s = 6$;
$a = (2, 8, 3, 7, 10)$, $s = 40$.

2.6. Antud järjendi elemendid kuni esimese nullini (erijuhul lõpuni) korrutada arvuga 2.

2.7. Loetleda kõik erinevad teekonnad $m \times n$ ruudustiku ülemisest vasakust tipust alumisse parempoolsesse tippu mööda ruutude servi, kus kõik liikumised toimuvad vaid alla ja paremale ning ühelgi etapil ei satuta algus- ja lõpppunkti ühendavast diagonaalist allapoole.

2.8. Vaheldumisi liita ja lahutada antud järjendi elemendid. Täpsemalt, arvutada aritmeetilise avaldise väärtus, milleks on vaheldumisi pluss- ja miinusmärgiga eraldatud järjendis loetletud arvud (negatiivsed sulgudes).

Näiteks, järjendi
$(0.5, -2.2, 3.8, 0.5, -0.3)$ korral arvutatakse $0.5+(-2.2)-3.8+0.5-(-0.3) = -4.7$.
Erijuhul, tühja järjendi $()$ ja ühe-elemendilise järjendi (a_0) puhul on tulemuseks vastavalt 0 ja a_0.

2.9. Leida antud järjendi positiivsete elementide kuupide summa.

Näiteks,
järjendi $(-2.0, 2.0)$ korral on tulemuseks 8.0,
järjendi $(2.0, 1.5, -18.0)$ korral aga 11.375.

2.10. Leida antud järjendi nullist erinevate elementide korrutis. (Null teguri korrutis loetakse võrdseks arvuga 1.0.)

Näiteks,
järjendi $(0.0, 4.0, 2.7)$ korral on tulemuseks 10.8 ja järjendi $(-2.0, 2.5)$ korral -5.0.

2.11. Antud on täisarvude järjendid a ja b. Leida kõikide selliste naturaalarvude summa, mis leiduvad nii järjendis a kui ka järjendis b.

Näiteks, $a = (1, 4, -3, 6, 1, 5, 7, 2, 2, 3, 5, 6)$ ja $b = (3, -4, -3, 7, 2, 2, 6, -3, 5)$ korral on tulemuseks 23 (naturaalarvude $6, 5, 7, 2, 3$ summa).

2.12. Arvutada antud järjendi a elementide kahekaupa korrutiste summa s, kus esimene korrutatakse viimasega, teine eelviimasega, kolmas lõpust kolmandaga jne

Paaritu arvu elementide korral keskmine element

(a) korrutatakse iseendaga (võetakse ruutu);

$$s = \sum_{i=1}^{\lceil n/2 \rceil} a_i a_{n-i+1}.$$

(b) keskmist elementi ei korrutata, kuid summas ta liidetakse.

Näiteid:
$a = (1, 2, 3, 4, 5, 6)$, $s = 28$ $(28 = 1 \cdot 6 + 2 \cdot 5 + 3 \cdot 4)$;
$a = (1, 2, 3, 4, 5)$,
variant (a) $s = 22$ $(22 = 1 \cdot 5 + 2 \cdot 4 + 3 \cdot 3)$;
variant (b) $s = 16$ $(16 = 1 \cdot 5 + 2 \cdot 4 + 3)$.

2.13. Antud järjendi $a = (a_1, a_2, \ldots, a_n)$ korral leida summa $s = \sum_{i=1}^{n} ia_i$.
Näiteks $a = (2,5,0,1,3)$ korral $s = 31 (31 = 1 \cdot 2 + 2 \cdot 5 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 3)$.

2.14. Leida antud järjendi

(a) (üks) pikim kasvav alamjärjend;

(b) suurim alamjärjendite summadest
(näiteks, järjendi $(-2,1,-3,4,-1,2,1,-5,4)$ alamjärjendite summadest suurim on 6);

(c) suurima summaga alamjärjend;

(d) pikim alamjärjend, mille kõik elemendid on paarikaupa ühistegurita.

(e) (üks) pikim kasvav osajärjend
(näiteks, järjendi $(2,1,-2,-3,4,-1,2,-5,4)$ pikim kasvav osajärjend on $(-3,-1,2,4)$.

2.15. Koostada antud järjendi a (pikkusega n) ja naturaalarvu m $(0 < m <= n)$ korral uus järjend b järgmisel viisil:

1) algselt $b = ()$;

2) järjendile b lisada järjendi a m-s element;

3) kuni kõik a elemendid ei ole järjendile b lisatud:
leida viimasena lisatud elemendist järjendis a tsükliliselt edasi lugedes m-s element järjendi a selliste elementide hulgast, mida pole veel järjendile b lisatud, ja lisada see järjendile b.

Näiteks,
$a = (1,2,3,4,5,6,7,8,9,10)$ ja $m = 2$ korral $b = (2,4,6,8,10,3,7,1,9,5)$.

2.16. Antud naturaalarvu n korral koostada paarikaupa erinevate juhuarvude järjend pikkusega $\lceil n/2 \rceil$, elementide väärtustega lõigult $[1;n]$.

2.17. Vaiga Vabariigis on lähenemas valimised. Vabariigis on kokku n parteid, kuid kuna vahetult enne valimisi keelustas parlament valimisliidud, ei jää parteidel edukaks esinemiseks muud üle, kui valimiste eel üksteisega liituda. Analüütikud on välja selgitanud, et kõige kasulikum on ühineda parteidel, mille liikmete arvud on lähedased. Niisiis ühinesidki kõigepealt need kaks parteid, mille liikmete arvud on kõige lähedasemad. Sellega kahanes parteide koguarv ühe võrra. Seejärel kordus ühinemisprotsess sama reegli järgi kuni valimistel osalemiseks jäi alles vaid kaks parteid A ja B.

Leida, millistest parteidest moodustusid liitparteid A ja B, kui on teada nii allesjäänud parteide A ja B liikmete arvud kui ka kõigi parteide liikmete arvud enne ühinemisi.

2.18. Loetleda antud paarisarvulise pikkusega järjendi elemendid ostsilleerivalt ümber keskkoha: kõigepealt keskkohast vasakult esimene element, siis keskkohast paremalt esimene, seejärel keskkohast vasakult teine, siis keskkohast paremalt teine jne.

Näiteks järjendi $(1,2,3,4,5,6,7,8,9,10)$ korral saadakse $(5,6,4,7,3,8,2,9,1,10)$.

2.19. Transponeerida antud järjend, st muuta elementide järjestus vastupidiseks.

2.20. Antud järjendis on õpilaste kontrolltöö hinded. Leida

(a) keskmine hinne;

(b) keskmisest madalama hindega õpilaste arv;

(c) maksimaalne hinne.

2.21. Antud järjendis on töötajate vanused. Leida

(a) iga töötaja kohta, mitu aastat on tal jäänud 65-aastaseks saamiseni;

(b) töötajate keskmine vanus;

(c) töötajate mediaanvanus;

(d) alla 40-aastaste töötajate arv;

(e) alla 40-aastaste töötajate keskmine vanus;

(f) minimaalne vanus.

2.22. Leida järjendi $(a_0,a_1,\ldots,a_n)$ selliste elementide summa, mis jaguvad 3-ga ja millede indeksid jaguvad samuti 3-ga.

Näiteks,
järjendi $(9,4,6,2,10,3,0,1,9,6)$ korral on selliseks summaks 15 $(=9+0+6)$.

2.23. Loetleda antud järjendi need elemendid, mis erinevad mingist täisruudust 5 võrra.

2.24. Leida antud järjendi suurima ja vähima elemendi vahe.

2.25. Leida antud järjendi kaks teineteisele suuruselt kõige lähemat elementi.

2.26. Leida suurim erinevus antud järjendi kahe naaberelemendi vahel.

2.27. Leida antud järjendi kaks elementi, mille suurim ühistegur on suurim.

2.28. Kontrollida, kas etteantavas järjendis leidub elemendipaar (x,y), kus $y=x^3$ või $x=y^3$.

2.29. Leida, millise väärtusega elemente leidub järjendis kõige rohkem. (Vastuseks võib olla mitu arvu.)

2.30. Kontrollida, kas järjendi elemendid on paarikaupa erinevad.

2.31. Antud on tasandi punktide koordinaadid $(x_1,y_1),\ldots,(x_n,y_n)$. Leida punktid, mis asuvad teineteisest kõige kaugemal.

2.32. Antud on järjend $a=(a_1,\ldots,a_n)$ ja täisarvujärjend $b=(b_1,\ldots,b_m)$. Arvutada summa

$$s=\sum_{j=1}^{m} f(a,b_j),$$

kus $f(a,i)=\begin{cases} a_i, \text{ kui } i\in[1,n], \\ 0, \text{ vastasel korral .} \end{cases}$

Teisisõnu, summeerida tuleb järjendi a elemendid, mille indeksid leiduvad järjendis b. Iga a elementi liita niimitu korda, kuimitu korda tema indeks esineb järjendis b (mh, kui a mingi elemendi indeksit järjendis b ei ole, siis seda ei liideta üldse).

Näiteks, kui $a=(2.5,-4.4,2.0,-1.5)$ ja $b=(2,6,1,-1,3,1)$,
siis $s=-4.4+2.5+2.0+2.5=2.6$.

2.33. Loetleda kahe järjendi selliste elementide väärtused, mida esineb mõlema järjendi elementide hulgas täpselt üks kord.

2.34. Antud kolme järjendi puhul moodustada nende ühiste elementide järjend.

Näiteks, järjendite
$(5,7,100,24,61,5)$,
$(1,61,5,5,10,40,24)$,
$(3,120,4,55,5,5,24,30,61)$
ühiste elementide järjend: $(5,24,61,5)$.

2.35. Kontrollida, kas antud järjendis leidub kolm arvu, mis sobivad täisnurkse kolmnurga küljepikkusteks.

2.36. Maamõõtja tegi kindlaks põllu rajamiseks ettenähtud n-nurkse maatüki nurkade koordinaadid riigi alusvõrgu teatava kindelpunkti O suhtes. Koordinaatide väärtusteks sai ta $(x_1,y_1),...,(x_n,y_n)$ ringkäigu järjestuses. Arvutada maatüki pindala.

Üldine võte kumera hulknurga pindala arvutamiseks on jagada see kolmnurkadeks: valida selle mingi tipp ja tõmmata sealt sirgjooned kõigisse teistesse tippudesse. Tekkinud kolmnurkade pindalad saab arvutada näiteks koordinaatidega määratud küljepikkuste järgi.

2.37. Leida antud järjendist arvukolmik(ud), milles suurima arvu ja vähima arvu vahe on minimaalne.

Näiteid:

- Järjendis $(1,6,4,0,5,9)$ on selleks kolmik $\{4,5,6\}$,
  kus $max - min = 6-4 = 2$. Muude arvukolmikute korral selline vahe ei ole kahest väiksem: nt $\{1,4,6\}$ korral 5, $\{0,1,4\}$ korral 4, $\{0,5,9\}$ korral 9.
- Järjendis $(6,4,5,4,6)$ leidub kaks minimaalse vahega $max-min=1$ arvukolmikut: $\{4,4,5\}$ ja $\{5,6,6\}$.

2.38. Leida (kui võimalik) antud järjendi $(a_0,a_1,\ldots,a_n)$ pikim alamjärjend, mis koosneb ainult arvu 2 astmetest.

Näiteks, järjendi $(2,4,3,1,2,4)$ korral on otsitavaks järjendi lõpus paiknev kolme-elemendiline alamjärjend (algusindeksiga 3 ja lõpuindeksiga 5).

2.39. Leida märgimuutude arv antud järjendis.

Näiteks, järjendis $(3,5,0,0,-2,-2,0,1,0,-2)$ on kolm märgmuutu.

2.40. Leida inversioonide arv antud järjendis (st mitmel korral suurema indeksiga element on väiksem kui mõni väiksema indeksiga element).

Näiteks, järjendis $(1,4,2,3)$ on kaks inversiooni, sest $2<4$ ja $3<4$.

2.41. Leida paarituarvulise pikkusega järjendi mediaan ilma järjendit sorteerimata.

2.42. Antud on järjend pikkusega n ja arv k, $1<k<n$. Antud järjendis paigutada väiksemad k elementi ümber järjendi algusesse, elementide väärtuste omavahelist järjestust säilitades nii uues (väiksemate) algusosas kui ka lõppu jäänud (suuremate) osas

(a) kui järjendi elementide väärtused on unikaalsed;

(b) kui järjendis võib olla korduvaid arve.

Näiteks (juhul a),

$$(3,5,8,4,2,6,7,1,9) \Rightarrow \begin{cases} (3,2,1,5,8,4,6,7,9), \text{ kui } k=3, \\ (3,5,4,2,6,1,8,7,9), \text{ kui } k=6. \end{cases}$$

2.43. Antud on järjend $a=(a_0,a_1,...,a_k)$, $k\geq 4$. Koostada viie-elemendiline järjend b, milles on järjendi a 2 suurimat ja 3 vähimat elementi (suvalises järjestuses).

2.44. Rivis seisab n inimest üksteise selja taga. Kui inimene on eesseisjast pikem, siis näeb ta lisaks eesseisjale veel teisi eespoolseid inimesi kuni selleni, kes on omakorda temast pikem.

Leida inimene, kes näeb rivis kõige kaugemale (kõige rohkem eesseisjaid), kui seisjate pikkused on antud n-elemendilise järjendina.

2.45. *Kolmnurkarvuks* nimetatakse naturaalarvu, mis esitub täisarvude 1 kuni n summana mingi naturaalarvu n korral (võib olla ka $n=0$). Kolmnurkarvude jada $k=(0,1,3,6,10,\ldots)$; $k_{i+1}=k_i+i$.

Leida antud järjendi $(a_0,a_1,\ldots,a_n)$ kolmnurkarvuliste indeksitega elementide summa.

Näiteks,
järjendi $(2,3,4)$ korral liidetakse elemendid indeksitega 0 ja 1, summaks saadakse 5 $(=2+3)$;
järjendi $(2,3,4,5)$ korral liidetakse elemendid indeksitega 0, 1 ja 3, summaks on 10 $(=2+3+5)$.

2.46. Antud järjend a salvestada uude järjendisse b keskelt väljapoole ümberjärjestatult: järjendi a keskmest ühekaugusel olevatest elementidest vasakpoolne eelneb parempoolsele järjendis b.
Näiteks,
$a=(1,2,3,4,5) \Rightarrow b=(3,2,4,1,5)$;
$a=(1,2,3,4,5,6) \Rightarrow b=(3,4,2,5,1,6)$.

2.47. Kontrollida, kas antud järjendis leidub kaks järjestikku paiknevat elementi, mille summa on samuti selle järjendi element.

2.48. Leida (kui võimalik) antud järjendi vähim selline indeks, millest väiksematel indeksitel paiknevate elementide summa võrdub ülejäänud elementide summaga.
Näiteks,
järjendi $(1,4,-1,2,-1,5)$ korral on selliseks indeksiks 2;
järjendi $(4,-3,2,-3)$ korral on selliseks indeksiks 0;
järjendi $(1,2,3,5,6)$ korral sellist indeksit ei ole.

2.49. Järjendisse a pikkusega n on kantud arvud hulgast $\{0,1,\ldots,n-1\}$, kus seejärel mõned neist on asendatud arvuga -1.
Näiteks, $a=(-1,-1,7,1,10,4,3,-1,-1,5,-1), n=11$.
Koostada järjend, mis on saadud sorteeritud järjendist $(0,1,\ldots,n-1)$ järjendis a puuduvate arvude asendamisel arvuga -1.
Ülaltoodud järjendi a korral on tulemuseks $(-1,1,-1,3,4,5,-1,7,-1,-1,10)$.

2.50. Antud järjendis paigutada elemendid ümber "kirjusse ritta" – negatiivsed ja positiivsed vaheldumisi (järjendi alguses).
Näiteks,
$(-5,-1,7,1,10,3,-2,-1,5,-6,4) \Rightarrow (-5,7.-1,1,-2,10,-1,3,-6,5,4)$.

2.51. Antud täisarvujärjendis paigutada ümber järjendi lõppu

(a) paaritud arvud;

(b) mittenegatiivsed arvud;

(c) nullid.

Näiteks,
(juht a) $(-1,0,-7,3,10,4,0,2,-1,-5,6) \Rightarrow (0,10,4,0,2,6,3,-1,-1,-5,-7)$;
(juht b) $(-1,0,-7,3,10,4,0,2,-1,-5,6) \Rightarrow (-1,-7,-1,-5,10,4,0,2,0,3,6)$;
(juht c) $[-1,0,-7,3,10,4,0,2,-1,-5,6] \Rightarrow (-1,-7,3,10,4,2,-1,-5,6,0,0)$.

2.52. Antud täisarvujärjendis a paigutada ümber järjendi lõppu

(a) paaritud arvud;

(b) mittenegatiivsed arvud.

Nii tulemuse algusosas (a – paarisarvud, b – negatiivsed arvud) kui ka lõpuosas (a – paaritud arvud, b – mittenegatiivsed arvud) peab olema säilitatud arvude esialgne järjestus järjendis a.

Näiteks,

(a) $(-1,0,-7,3,10,4,0,2,-1,-5,6) \Rightarrow (0,10,4,0,2,6,-1,-7,3,-1,-5)$;

(b) $(-1,0,-7,3,10,4,0,2,-1,-5,6) \Rightarrow (-1,-7,-1,-5,0,3,10,4,0,2,6)$.

2.53. Antud järjendis $a = (a_0, a_1, \ldots, a_n)$ paigutada elemendid ümber selliselt, et

$a_{i-1} >= a_i$, kui i on paarisarv, $a_{i-1} <= a_i$, kui i on paaritu ($i = 1,2,\ldots,n$).

2.54. Antud järjendis $a = (a_0, a_1, \ldots, a_n)$, $n \geq 3$, paigutada elemendid ümber selliselt, et paarisindeksitega osajärjend $(a_0, a_2, \ldots)$ oleks mittekahanev ja paarituindeksitega osajärjend $(a_1, a_3, \ldots)$ oleks mittekasvav.

Näiteks, $(-1,8,7,1,10,4,0,2,-1,5,6) \Rightarrow (-1,10,-1,8,0,7,1,6,2,5,4)$.

2.55. Antud on arvude järjend $a = (a_0, a_1, \ldots, a_n)$ ja selle indeksite hulga $\{0,1,\ldots,n\}$ mingi permutatsioon p. Järjend a tuleb ümber järjestada vastavalt permutatsioonile p, st arv järjendi a kohalt i tuleb salvestada kohale p_i ($i = 0,1,\ldots,n$).

Näiteks,

$a = (25,13,56,44,75,33)$,

$p = (3,0,4,1,2,5)$.

Ümberjärjestatud $a = (13,44,75,25,56,33)$.

2.56. Antud järjendi iga element asendada temale eelneva ja temale järgneva elementide korrutisega. Puuduv eelmine ja puuduv järglane lugeda arvuks 1.

Näiteks, $(-1,0,-7,3,10,4,0,2,-1,-5,6) \Rightarrow (0,7,0,-70,12,0,8,0,-10,-6,-5)$.

2.57. Antud bitijärjendis paigutada nullid järjendi algusesse.

Näiteks, $(1,0,1,0,1,1,1,0,0,1,0) \Rightarrow (0,0,0,0,0,1,1,1,1,1,1)$.

2.58. Antud järjendi iga element (va viimane) asendada suurimaga temale järgnevate elementide seas.

Näiteks, $(99,10,20,43,35,19,12,31,0,1) \Rightarrow (43,43,43,35,31,31,31,1,1,1)$.

2.59. Antud bitijärjendis asendada üks nullidest ühega ($0 \to 1$), nii et moodustuks võimalikult pikk ühtedest koosnev alamjärjend. Näiteks,

$(1,0,0,0,1,1,0,\underline{0},1,1,1,0) \Rightarrow (1,0,0,0,1,1,0,\underline{1},1,1,1,0)$;

$(1,0,1,\underline{0},1,1,1,0,0,1,1,1,0) \Rightarrow (1,0,1,\underline{1},1,1,1,0,0,1,1,1,0)$.

2.60. Antud korduvate elementideta järjend $a = (a_0, a_1, \ldots, a_n)$ korraldada ümber siksakiliseks, st nii, et iga naabrite paari (a_i, a_{i+1}) $(i = 0, 1, \ldots, n-1)$ korral
$a_i < a_{i+1}$, kui i on paarisarv,
$a_i > a_{i+1}$, kui i on paaritu.
Näiteks, $(9,5,16,13,10,4,2) \Rightarrow (5,16,9,13,4,10,2)$.

2.61. Antud on järjend ja täisarv k.
Antud järjendis nihutada elemendid $|k|$ kohta tsükliliselt

- paremale, kui $k > 0$;
- vasakule, kui $k < 0$.

Näiteks, järjendi $(3,7,6,0,3)$ korral,
kui $k = 3$, saadakse $(6,0,3,3,7)$;
kui $k = -3$, saadakse $(0,3,3,7,6)$;
kui $k = -7$, saadakse $(6,0,3,3,7)$.

2.62. Antud järjendis nihutada elemendid $|k|$ kohta tsükliliselt

- paremale, kui $k > 0$;
- vasakule, kui $k < 0$,

kasutades ainult järjendi või selle osade transponeerimist.

2.63. Antud on arv x ja järjend a, mis on saadud mingi sorteeritud järjendi tsüklilisel nihutamisel mingi arv kohti. Kontrollida, kas selles järjendis leidub

(a) arv x;

(b) arvupaar summaga x.

2.64. Lähtudes neljakohalisest naturaalarvust n, mille numbrid ei ole kõik samad (st mitte $1111, 2222, \ldots, 9999$), sooritatakse korduvalt järgmised tehted:

1. $a := arv$, mis saadud arvu n numbrite järjestamisel mittekasvavalt;
2. $b := arv$, mis saadud arvu n numbrite järjestamisel mittekahanevalt;
3. $n := a - b$;
4. kui $n = 6174$ (nn *Kaprekari konstant*), siis lõpetada.

On teada, et selle konstandini jõutakse alati ülimalt 7 sammuga. Järgmised tehted enam n väärtust (6174) ei muudaks, sest $7641 - 1467 = 6174$.

Antud on neljakohaline naturaalarv n (mitte $1111, 2222, \ldots, 9999$). Loetleda pärast igat 3. sammu saadavad arvukolmikud (a,b,n).
Näiteks $n = 6317$ korral on nendeks
$(7631, 1367, 6264)$,
$(6642, 2466, 4176)$,
$(7641, 1467, 6174)$.

2.65. Antud n-elemendilise järjendi i-ndaks elemendiks on $((-1)^{i+1})/(2*i-1)$, $i=1,\ldots,n$.
Leida selle järjendi neljakordne summa.

2.66. Antud järjendis asendada iga element summaga järjendi algusest kuni selle elemendini (kaasa arvatud), kusjuures jooksvale elemendile liidetakse

(a) eelnevate elementide esialgsed väärtused;

(b) eelnevate elementide uued (juba muudetud) väärtused.

2.67. Antud on järjend $a=(a_0,a_1,\ldots,a_n)$ ja üks indekseid i sellel. Leida (kui võimalik) elemendile a_i lähim (järjendis eelnev või järgnev) temast suurem element.
Näiteks, järjendi $a=(1,4,3,2,5,7)$ korral,
kui $i=1$, $a_1=4$, siis leitakse $a_4=5$;
kui $i=2$, $a_2=3$ siis leitakse $a_1=4$.

2.68. Antud on täisarvude järjend $(a_0,a_1,\ldots,a_n)$ ja algarv m. Arvutada selle järjendi kontrollsumma ks baasil m:
$ks:=0$;
$\forall i, i=0,1,\ldots,n:$
$\quad ks:=((ks+a_i)*m) \mod 10000007$;

2.69. Funktsiooni $f(x)$ väärtused lõigul $[a;b]$ sammuga $h=(b-a)/n$ on salvestatud järjendisse y: $y_1=f(a), y_2=f(a+h), y_3=f(a+2\cdot h),\ldots,y_n=f(b)$.
Asendada järjendis y iga element $y_i (i=2,3,\ldots,y_{n-1})$ kolmiku y_{i-1},y_i,y_{i+1} aritmeetilise keskmisega. (Sellega muudetaks $y=f(x)$ graafik sujuvamaks, vähem "sakiliseks".)

2.70. Loetleda antud kordarvu algtegurid mittekahanevalt.
Näiteks,
arvu 1650 algtegurid on $2,3,5,5,11$; arvu 1653 algtegurid on $3,19,29$.

2.71. Koostada n-elemendiline järjend $(a_0,\ldots,a_{n-1})$, kus a_i $(i=0,1,\ldots,n-1)$ on esimese $(i+2)$-ga jaguva Fibonacci arvu järjekorranumber.
Näiteks, $n=10$ korral on tulemuseks $[3,4,6,5,12,8,6,12,15,10]$.
(Esimesed Fibonacci arvud on teatavasti 0)0, 1)1, 2)1, 3)2, 4)3, 5)5, 6)8, 7)13, 8)21 9)34, 10)55, 11)89, 12)144, 13)233, 14)377, 15)610, 16)987, ...)

2.72. Antud on m ja n, $1<m<n$. Koostada n-elemendiline järjend $a=(a_0,\ldots,a_{n-1})$, milles esimesed m elementi on kahe astmed,
$a_i=2^i$, $i=0,1,\ldots,m-1$
ja iga järgmine on talle eelneva m elemendi summa,
$a_j=a_{j-1}+a_{j-2}+\ldots+a_{j-m}$, $j=m,m+1,\ldots,n-1$.
Näiteks, $m=4$ ja $n=10$ korral
koostatakse järjend $(1,2,4,8,15,29,56,108,208,401)$
(elementide indeksitega $0,1,2,3,\underline{4},5,6,7,8,9$).

2.73. Koondada antud järjend

(a) nihutamisega vasakule;

(b) nihutamisega paremale.

Koondamine: järjendis olnud iga võrdsete naabrite paari korral asendatakse üks paari liige nulliga ja teine korrutatakse kahega; seejärel paigutatakse kõik järjendisse jäänud mittenullid ümber järjendi algusesse (juhul a) või lõppu (juhul b).

Näiteks (nihutamisega vasakule):
$(1,3,3,3,3) \Rightarrow (1,6,6,0,0)$;
$(0,32,3,3,6,0,3,0,3,0) \Rightarrow (32,6,6,3,3,0,0,0,0,0)$.

2.74. Antud bitijärjendi korral leida selliste erinevate osajärjendite arv, mis koosnevad ainult ühesugustest bittidest (kas ainult ühtedest või ainult nullidest) ja milles on paarisarv elemente.

Näiteks,
bitijärjendi $(0,0,1,0,1,0,1,1,1,0,1,1)$ korral on tulemuseks arv 5, sest erinevateks nö lubatud osajärjenditeks on $(0,0)$,$(0,0,0,0)$,$(1,1)$,$(1,1,1,1)$,$(1,1,1,1,1,1)$.

2.75. Antud on n-elemendiline järjend. Leida selle järjendi suurima kõigi elementide korrutisega

(a) alamjärjend, kui $n < 50$,

alamjärjend pikkusega 4, kui $n \geq 50$;

(b) osajärjend, kui $n < 15$,

osajärjend pikkusega 4, kui $n \geq 15$.

Näiteks (juht b), 17-elemendilises järjendis
$(1.8, 0.0, 1.2, -0.8, 0.7, -0.5, -1.9, -0.7, 1.3, 1.0, -0.3, -1.2, 2.2, 0.1, 0.2, -0.3, -2.5)$
on suurima korrutisega (18.81) 4-elemendiliseks osajärjendiks
$(1.8, -1.9, 2.2, -2.5)$.

II Muutuva pikkusega ehk dünaamilised järjendid

2.76. Antud on sorteeritud järjendid a ja b. Kanda järjendi b kõik elemendid üle järjendisse a, viimase sorteeritust säilitades.

Näiteks, $a = (-1,0,2,4,5,7,9)$, $b = (1,1,2,3,8,16)$;
pärast b elementide ülekandmist a-sse $a = (-1,0,1,1,2,2,3,4,5,7,8,9,16)$, $b = ()$.

2.77. Antud järjendist a eemaldada korduvad elemendid, säilitades allesjäänute omavahelise järjestuse

(a) kui a on sorteeritud;

(b) kui a on sorteerimata.

2.78. Antud on järjendid a ja b. Järjendist a eemaldada need elemendid, mis ei esine järjendis b.

Näiteks, $a = (-1,9,4,2,0,7,9)$, $b = (1,1,8,3,2,9)$; tulemus: $a = (9,2,9)$.

2.79. Tähistame $I(n) = (0,1,\ldots,n-1)$ – mittenegatiivsete järjestikuste täisarvude järjend pikkusega n.

Olgu antud n ja $I(n)$ osajärjend b. Koostada järjend a, mis on saadud järjendist $I(n)$ järjendisse b kuuluvate arvude eemaldamisel.

Näiteks,
$n = 16$, $b = (0,1,7,9,13,14,15) \Rightarrow a = (2,3,4,5,6,8,10,11,12)$.

2.80. Antud on unikaalsete elementide järjend a ja selle üks osajärjend a'. Koostada järjend a'', mis on saadud järjendist a järjendi a' sümbolite eemaldamisel.

Näiteks,
$a = (11,8,2,-3,0,5,4)$, $a' = (8,0,5) \Rightarrow a'' = (11,2,-3,4)$.

2.81. Naturaalarvuga n tehakse järgmisi operatsioone seni, kuni n-i väärtuseks saab 1:

- kui n jagub 2-ga, siis jagatakse ta 2-ga;
- kui n ei jagu 2-ga, siis asendatakse ta arvuga $3n+1$.

Collatzi hüpotees väidab, et ükskõik millisest naturaalarvust lähtudes jõuame alati arvuni 1.

Antud on naturaalarv n. Koostada ülalkirjeldatud operatsioonide käigus tekkiv arvujärjend $n,\ldots,1$.

2.82. Loetleda kõik lõigul $[1;500]$ leiduvad täiuslikud arvud (vt ülesanne 1.20).

2.83. Loetleda antud järjendi unikaalsed elemendid (elemendid, mis esinevad selles järjendis täpselt üks kord).

2.84. Antud on järjend a ja täisarvujärjend b. Loetleda antud järjendi a sellised elemendid, mille indeks esineb järjendis b.

2.85. Leida (kui võimalik) antud järjendis pikim vähemalt kolme-elemendiline osajärjend, mis on aritmeetiline jada.

2.86. Leida järjendi $(a_0,a_1,\ldots,a_n)$ selliste elementide indeksid, millest tahapoole (indeksite mõttes) jääb kõige rohkem temast suuremaid elemente.

Näiteks, järjendi $(16,15,16,14,14,14,19,18,15)$ korral on tulemuseks indeksite järjend $(1,3,4,5)$.

2.87. Ajateenistuses olles sõnastas matemaatikatudeng Juku ülesande, milleks sai inspiratsiooni jao igahommikusest jooksuharjutusele kogunemisest. Nimelt oli jaoülemal kombeks käsutada sõdurid esmalt suvalises järjestuses rivvi, võrdsete vahedega üksteise selja taha. Igal hommikul nägi Juku üldiselt erinevat arvu eesseisjate vormimütse. Sellest tuli Jukul mõte formuleerida järgmine ülesanne.

Antud on ülesrivistatud sõdurite pikkuste järjend y. Sõdureid kujutame x-teljele paigutatud võrdsete vahedega püstlõikudena, i-nda püstlõigu pikkus on y_i. Tähistame $k(i,j)$ – sirge tõus, mis läbib i-nda ja j-nda püstlõigu ülemisi otsi. Vaatleme kahte püstlõiku (sõdurit) nr i ja j $(i < j)$. Sätestame, et i-s sõdur näeb j-nda sõduri mütsi, kui $k(i,j) > k(i,r)$ iga $r = i+1, i+2, \ldots, j-1$ korral. Iga sõduri korral tuleb leida, mitme eesseisja vormimütsi ta näeb.

2.88. Vaadeldavas positiivsete täisarvude jadas $u_0, u_1, \ldots$ kehtib iga $i = 0, 1, \ldots$ korral:

$$u_{i+1} = \begin{cases} \frac{1}{2}u_i & \text{kui } u_i \text{ on paarisarv,} \\ a + u_i & \text{kui } u_i \text{ on paaritu,} \end{cases}$$

kus $a > 0$ on paarituarvuline konstant.

Saab tõestada, et see jada on alates mingist kohast perioodiline.

Leida selle jada perioodi pikkus ning jada liikme indeks, millest periood algab.

2.89. Antud on k sorteeritud järjendit ($k \geq 0$). Koostada nende ühildamise (põimimise) teel saadav sorteeritud järjend.

2.90. Antud järjendi $(a_0, a_1, \ldots, a_n)$ jaoks koostada indeksite $(0, 1, \ldots, n)$ nn sorteeriv permutatsioon $(p_0, p_1, \ldots, p_n)$, nii et $a_{p_{i-1}} <= a_{p_i}$, iga $i = 1, \ldots, n$ korral.

Näiteks, järjendi $(3,2,4,2,5,9)$ (sorteeritult $(2,2,3,4,5,9)$) sorteeriv permutatsioon on $(1,3,0,2,4,5)$.

3. Maatriksite töötlemine

3.1. Antud on $n \geq 5$. Koostada (rida-realt) sümbolite maatriks, milles esinevad ainult sümbolid '#' ja ' ' (tühik) ning mille ridade kaupa esitamisel tekib kujund (näidetes $n = 5$)

(a) ristkülik $n \times n$ maatriksina, nt

```
#####
#   #
#   #
#   #
#####
```

(b) rööplülik $n \times (2n-1)$ maatriksina, nt

```
#####
 #   #
  #   #
   #   #
    #####
```

(c) langev trepp $n \times \frac{n(n+1)}{2}$ maatriksina, nt

```
#
 ##
  ###
   ####
    #####
```

(d) tõusev trepp $n \times \frac{n(n+1)}{2}$ maatriksina, nt

```
    #
   ##
  ###
 ####
#####
```

(e) kallutatud romb $(2n-1) \times n$ maatriksina, nt

```
   #
  ##
 # #
#  #
#   #
#  #
# #
##
#
```

(f) romb $(2n-1)\times(2n-1)$) maatriksina, nt

```
   #
  # #
 #   #
#     #
#     #
 #   #
  # #
   #
```

(g) telk $n\times(4n-3)$ maatriksina, nt

```
     #
   ## ##
  ###   ###
 ####     ####
#####       #####
```

(h) püramiid $n\times(2n-1)$ maatriksina, nt

```
    #
   # #
  # # #
 # # # #
# # # # #
```

(i) spiraal $n\times n$ maatriksina, nt

```
#####
    #
### #
#   #
#####
```

Märkus. Järnevates ülesannetes käsitletakse ainult arvude maatrikseid.

3.2. Koostada 9×9 maatriks A, milles $A_{ij}=i\cdot j$ ($i=1,\ldots,9$ ja $j=1,\ldots,9$).

3.3. Koostada 10×2 maatriks A, milles

$$A_{i1}=i$$

$$A_{i2}=\sum_{n=1}^{i}\frac{1}{n}$$

($i=1,\ldots,10$).

3.4. Transponeerida antud ruutmaatriks.

3.5. Koostada antud maatriksi transponeeritud maatriks.

3.6. Antud maatriksi korral koostada uus maatriks, milles on vahetatud kaks etteantud indeksitega

(a) rida;

(b) veergu.

3.7. Leida antud maatriksi

(a) peadiagonaali elementide summa;

(b) kõrvaldiagonaali elementide summa;

(c) peadiagonaalist allpool asuvate elementide summa.

3.8. Antud on $m \times n$ maatriks A. Koostada maatriks mõõtmetega

- $m \times (n-1)$, kui $m \neq n$,
- $(m-1) \times m$, kui $m = n$,

mis saadakse maatriksist A

(a) peadiagonaali elementide kustutamisel;

(b) kõrvaldiagonaali elementide kustutamisel;

(c) juhuslikult valitud m elemendi kustutamisel.

3.9. Koostada maatriks, mis saadakse antud maatriksi

(a) 90^0 kraadi päripäeva pööramise teel;

(b) 90^0 kraadi vastupäeva pööramise teel.

3.10. Loetleda antud maatriksi elemendid ussimustrijärjestuses, mis algab

(a) maatriksi vasakust ülemisest nurgast suunaga paremale ning põrkel vastu maatriksi serva jätkab alati järgmisest reast (kui see leidub) vastassuunaliselt;

(b) maatriksi vasakust ülemisest nurgast suunaga alla ning põrkel vastu maatriksi serva jätkab alati järgmisest veerust (kui see leidub) vastassuunaliselt.

Näiteks, maatriksi $\begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 4 & 5 & 6 \\ 0 & 7 & 8 & 9 \end{pmatrix}$ elemendid ussimustrijärjestuses:

a) $0,1,2,3,6,5,4,0,0,7,8,9$;

b) $0,0,0,7,4,1,2,5,8,9,6,3$.

3.11. Loetleda antud maatriksi kõik sadulpunktid.

(Maatriksi *sadulpunktiks* on sellise elemendi lahter, mis on oma väärtuselt suurim veerus ja vähim reas või suurim reas ja vähim veerus.)

3.12. Antud maatriksi korral koostada 4-realine maatriks, milles ühe rea elementideks on antud maatriksi ühe diagonaali elemendid (ülemistest nurkadest ülalt-alla ning alumistest nurkadest alt-üles).

Näiteks, $\begin{pmatrix} 3 & 5 & 0 & 2 & 2 & 4 & 8 & 0 \\ 1 & 3 & 1 & 3 & 5 & 8 & 8 & 1 \\ 1 & 6 & 4 & 2 & 6 & 1 & 5 & 3 \\ 3 & 0 & 8 & 3 & 3 & 9 & 9 & 1 \\ 4 & 3 & 1 & 2 & 5 & 7 & 0 & 7 \end{pmatrix} \Rightarrow \begin{pmatrix} 3 & 3 & 4 & 3 & 5 \\ 4 & 0 & 4 & 3 & 2 \\ 7 & 9 & 1 & 5 & 2 \\ 0 & 8 & 1 & 3 & 2 \end{pmatrix}$.

3.13.
Leida antud maatriksi sellise rea number, mille elementide summa on vähim.

3.14. Leida antud maatriksi

(a) ülalt-alla diagonaalide summad;

(b) nende elementide summa, mis asuvad ülalt-alla diagonaalil.

Näiteks, maatriksi $\begin{pmatrix} 3 & 3 & 4 & 3 & 5 \\ 4 & 0 & 4 & 3 & 2 \\ 7 & 9 & 1 & 5 & 2 \\ 0 & 8 & 1 & 3 & 2 \end{pmatrix}$ korral on vastusteks

a) 7 ja 17 $(3+0+1+3)$ ja $(5+3+1+8)$;

b) 23 $(3+0+1+3+5+3+8)$.

3.15. Leida antud maatriksi nende kahe rea numbrid, mille skalaarkorrutis on vähim.

3.16. Antud m-realise maatriksi korral koostada m-realine tabel, mille i-ndas reas on loetletud need maatriksi i-nda rea elemendid, mida selle maatriksi teistes ridades ei esine ($i = 1, 2, \ldots, m$).

3.17. Koostada maatriks, mis on antud maatriksi peegelduseks

(a) vertikaalse kesktelje suhtes;

(b) horisontaalse kesktelje suhtes.

3.18. Antud on maatriks ning selle mingi rea number i ja mingi veeru number j, Koostada maatriks, mis on saadud antud maatriksist i-nda rea ja j-nda veeru kustutamise teel.

3.19. Koostada maatriks, mis saadakse antud maatriksi ridade ühe võrra ülespoole tsüklilise nihutamise teel.

3.20. Loetleda antud maatriksi elemendid päripäeva (kahanevat) spiraali mööda.

Näiteks, maatriksi $\begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 4 & 5 & 6 \\ 0 & 7 & 8 & 9 \end{pmatrix}$ korral on loeteluks $0, 1, 2, 3, 6, 9, 8, 7, 0, 0, 4, 5$.

3.21.
Leida antud maatriksi suurima elementide kogusummaga

(a) üks 2×2 alammaatriks ja selliste arv;

(b) üks alammaatriks ja selliste arv.

3.22. Koostada maatriks, mis on saadud antud maatriksist ridade sorteerimise teel reasummade järgi mittekahanevalt.

3.23. Koostada maatriks, mis on antud ruutmaatriksi peegelduseks

(a) peadiagonaali suhtes;

(b) kõrvaldiagonaali suhtes.

3.24. Kontrollida, kas antud maatriksi peadiagonaal domineerib ridade üle, st kas igas reas on peadiagonaali elemendi absoluutväärtus suurem kui ülejäänud samas reas asuvate elementide absoluutväärtuste summa.

3.25.
Vaatleme "pikendatud diagonaale", mis maatriksi ääre vastu põrgates jätkuvad vastavalt peegeldumisseadustele. Näiteks maatriksis mõõtmetega 3×5 nurgast $(1,1)$ algav diagonaal läbib (pikendatuna) järjest elemendid indeksitega $(1,1)$, $(2,2)$, $(3,3)$, $(2,4)$, $(1,5)$, $(2,4)$, $(3,3)$, $(2,2)$, $(1,1)$.

Olgu antud maatriks, milles nii ridu kui ka veerge on vähemalt 2. Leida kõik elemendid, mis paiknevad antud maatriksi vasakust ülemisest nurgast algaval pikendatud diagonaalil kuni esimese põrkumiseni maatriksi vasaku äärega.

Näiteks maatriksis

$$\begin{pmatrix} 8 & 8 & 7 & 9 & 1 \\ 2 & 1 & 8 & 7 & 6 \\ 4 & 9 & 5 & 3 & 2 \\ 9 & 5 & 8 & 4 & 5 \\ 6 & 2 & 7 & 6 & 8 \\ 3 & 8 & 8 & 7 & 1 \\ 7 & 3 & 3 & 4 & 9 \\ 8 & 0 & 0 & 9 & 7 \end{pmatrix}$$

diagonaalteekond (pikendatud diagonaal) vasakust ülemisest nurgast tagasi esimesse veergu on 8, 1, 5, 4, 8, 7, 3, 0, 7.

3.26.
Olgu antud arvude maatriks A. Koostada maatriks B, mille ridadeks on need A read, mille kolmnurkarvulise indeksiga kohtadel asuvate elementide summa on suurem kui ühegi eelmise rea kolmnurkarvulise indeksiga kohtadel asuvate elementide summa. Maatriksi B esimeseks reaks olgu A esimene rida. (*Kolmnurkarv*: vt ülesanne 2.45.)

Näiteks,

$$A = \begin{pmatrix} 2 & -3 & 1 \\ 2 & 3 & 4 \\ 1 & 4 & -7 \\ 0 & 0 & 1 \\ 8 & 8 & 8 \end{pmatrix}$$

$$B = \begin{pmatrix} 2 & -3 & 1 \\ 2 & 3 & 4 \\ 8 & 8 & 8 \end{pmatrix}$$

3.27. Kontrollida, kas antud täisarvude maatriksis leidub kaks vahetult teineteise all paiknevat rida, millest alumise kõik arvud jaguvad vastavate ülemise rea arvudega (sh null loetakse nulliga jaguvaks, nullist erinevad arvud ei jagu nulliga).

Näiteks,

maatriksis $\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 \\ 0 & 2 & 4 & 12 \end{pmatrix}$ sellised read leiduvad,

maatriksis $\begin{pmatrix} 1 & 3 \\ 2 & 4 \\ 3 & 5 \\ 4 & 8 \end{pmatrix}$ selliseid ridu ei ole.

3.28. Koostada ruutmaatriks, mille esimeseks reaks on antud järjend, igaks järgnevaks reaks aga eelmine rida tsükliliselt ühe elemendi võrra paremale nihutatult.

Näiteks,

$$(1,2,3,4) \Rightarrow \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{pmatrix}$$

3.29. Koostada ruutmaatriks, mille saab antud maatriksist kas vasakpoolse või ülemise sobiva laiusega riba eemaldamisel.

Näiteks,

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 5 & 2 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 5 & 2 & 1 \end{pmatrix}; \qquad \begin{pmatrix} 1 & 3 & 5 & 7 & 9 \\ 0 & 2 & 4 & 6 & 8 \end{pmatrix} \Rightarrow \begin{pmatrix} 7 & 9 \\ 6 & 8 \end{pmatrix}.$$

3.30. Olgu antud $m \times n$ maatriks $A = (a_{ij})$ $(m > 0, n > 0)$. Konstrueerida

(a) maatriks $B = (b_{ij})$, mis on saadud maatriksist A peegeldusena horisontaalse kesktelje suhtes;

(b) $m \times 2n$ maatriks C, mille i-ndaks reaks on maatriksite A ja B i-ndate ridade konkatenatsioon $a_{i1}\ a_{i2} \ldots a_{in}\ b_{i1}\ b_{i2} \ldots b_{in}$, $\quad i = 1,2,\ldots,m$.

Näiteks,

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 6 \\ 3 & 4 & 7 & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 3 & 4 & 7 & 0 \\ 2 & 3 & 4 & 6 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 2 & 3 & 4 & 3 & 4 & 7 & 0 \\ 2 & 3 & 4 & 6 & 2 & 3 & 4 & 6 \\ 3 & 4 & 7 & 0 & 1 & 2 & 3 & 4 \end{pmatrix}$$

3.31. Olgu antud maatriks A ja selle ühe elemendi indeksid i ja j. Koostada 4-elemendiline järjend b, milles on arv a_{ij} ja A need elemendid, mis paiknevad antud elemendi suhtes vertikaaltelje, keskpunkti ning horisontaaltelje suhtes sümmeetriliselt (selles järjestuses).

$$A = \begin{pmatrix} 8 & 5 & 7 & 9 & 5 & 6 & 1 & 4 \\ 8 & 5 & 4 & 9 & 3 & 5 & 1 & 3 \\ 7 & 4 & \mathbf{8} & 4 & 9 & \mathbf{3} & 2 & 1 \\ 5 & 2 & \mathbf{9} & 5 & 7 & \mathbf{4} & 3 & 9 \\ 9 & 6 & 8 & 0 & 1 & 0 & 1 & 1 \\ 6 & 7 & 8 & 1 & 2 & 9 & 3 & 9 \end{pmatrix}$$

Kui $i, j = 3,3$, siis $b = (8,3,4,9)$.

3.32. Olgu antud maatriks A ja selle ühe, A ülemises vasakpoolses veerandis asuva elemendi a_{ij} indeksid i ja j. Koostada järjend b, milles on järgmisel teekonnal esinevad arvud:

- antud elemendist a_{ij} kuni temast vertikaaltelje suhtes sümmeetrilise elemendini;
- viimasest edasi kuni keskpunkti suhtes sümmeetrilise elemendini;
- viimasest edasi kuni horisontaaltelje suhtes sümmeetrilise elemendini;
- viimasest edasi kuni algse elemendini a_{ij}.

Sama elementi koostatavasse järjendisse korduvalt ei lisata.

Näiteks:

$$A = \begin{pmatrix} 8 & 5 & 7 & 9 & 5 & 6 & 1 & 4 \\ 8 & 5 & 4 & 9 & 3 & 5 & 1 & 3 \\ 7 & 4 & \mathbf{8} & \mathbf{4} & \mathbf{9} & \mathbf{3} & 2 & 1 \\ 5 & 2 & \mathbf{9} & \mathbf{5} & \mathbf{7} & \mathbf{4} & 3 & 9 \\ 9 & 6 & 8 & 0 & 1 & 0 & 1 & 1 \\ 6 & 7 & 8 & 1 & 2 & 9 & 3 & 9 \end{pmatrix}$$

Kui $i, j = 3,3$, siis $b = (8,4,9,3,4,7,5,9)$.

3.33. Koostada antud maatriksi keskmistes lahtrites olevate elementide järjend.

Lahtrit loeme keskmiseks, kui temast reas vasakule ja paremale jäävate lahtrite arvud erinevad ülimalt 1 võrra ning ka temast veerus üles ja alla jäävate lahtrite arvud erinevad ülimalt 1 võrra. Koostatud järjendi pikkuseks on antud maatriksi keskmiste lahtrite arv.

Näiteks,

maatriksi $\begin{pmatrix} 9 & 2 & 4 & 0 \\ 2 & \mathbf{7} & \mathbf{2} & 5 \\ 3 & 0 & 9 & 8 \end{pmatrix}$ korral on tulemusjärjendiks $(7,2)$;

maatriksi $\begin{pmatrix} 2 & 1 & 6 & 5 \\ 5 & \mathbf{4} & \mathbf{3} & 0 \\ 2 & \mathbf{0} & \mathbf{9} & 8 \\ 1 & 1 & 6 & 9 \end{pmatrix}$ korral on tulemusjärjendiks $(4,3,0,9)$;

maatriksi $\begin{pmatrix} 2 & 5 & 6 \\ 7 & \mathbf{9} & 2 \\ 1 & 4 & 3 \end{pmatrix}$ korral on tulemusjärjendiks (9).

3.34. Koostada antud maatriksi kõik peadiagonaaliga samasihilistel diagonaalidel olevate elementide järjendid, vasakpoolseimast diagonaalist parempoolseimani.

Näiteks:

maatriksi $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{pmatrix}$ korral on tulemusjärjenditeks

(2)
$(1,4)$
$(1,3,6)$
$(2,5,8)$
$(3,7)$
(4)

maatriksi $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \\ 4 & 8 & 0 \end{pmatrix}$ korral on tulemusjärjenditeks

(4)
$(3,8)$
$(2,6,0)$
$(1,4,9)$
$(2,6)$
(3)

3.35. Antud on $m \times n$ maatriks A ja indeksijärjendi $(1,2,\ldots,m)$ osajärjend I. Koostada maatriks B, kuhu on võetud maatriksi A need read, mille number ei leidu antud järjendis I.

Näiteks,

maatriksi $A = \begin{pmatrix} 9 & 2 & 4 \\ 3 & 0 & 9 \\ 1 & 2 & 3 \\ 2 & 4 & 6 \end{pmatrix}$ ja $I = (2,3)$ korral $B = \begin{pmatrix} 9 & 2 & 4 \\ 2 & 4 & 6 \end{pmatrix}$;

maatriksi $A = \begin{pmatrix} 1 & 3 & 5 & 9 \\ 6 & 7 & 8 & 9 \\ 9 & 0 & 8 & 7 \end{pmatrix}$ ja $I = (1,3)$ korral $B = \begin{pmatrix} 6 & 7 & 8 & 9 \end{pmatrix}$.

3.36. Paigutada antud järjendi elemendid tabelina nii, et tabeli veergude arv oleks võrdne ridade arvuga ja võimalikult väike. Tabel tuleb täita ridade kaupa (vasakult paremale, ülalt alla) järjendi elementidega nende järjestust muutmata. Kui massiivi elemendid saavad otsa, võivad tabeli viimased read lühemaks või tühjaks jääda.

Näiteks,

$$(5,3,1,3,5,6) \Rightarrow \begin{pmatrix} 5 & 3 & 1 \\ 3 & 5 & 6 \end{pmatrix};$$

$$(1,2,3,4,5,6,7) \Rightarrow \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & & \end{pmatrix};$$

$$(9,8,7,6,5,4,3,2,1,0) \Rightarrow \begin{pmatrix} 9 & 8 & 7 & 6 \\ 5 & 4 & 3 & 2 \\ 1 & 0 & & \end{pmatrix}.$$

3.37. Leida antud täisarvude maatriksi peadiagonaali (st ülemisest vasakust nurgast algava diagonaali) elementide korrutis.

Näiteks,

$$A = \begin{pmatrix} 2 & 5 & 4 \\ 3 & 1 & 6 \\ 5 & 5 & 4 \\ 1 & 9 & 9 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

Maatriksi A peadiagonaali elementide korrutis on 8, maatriksi B peadiagonaali elementide korrutis on 6.

3.38. Maatriksi ridadeks on paarikaupa erinevad reaalarvukolmikud. Iga kolmiku esimesed kaks arvu määravad tasandile joonestatud ruudu keskpunkti, kolmas arv aga on selle ruudu küljepikkus. Ruutude küljed on rööbiti koordinaattelgedega. Ruudud võivad (osaliselt) kattuda ning mõni piirkond tasandil võib olla kaetud kahe-, kolme- ja enamakordselt.

Leida ruutudega kaetud tasandiosa pindala.

3.39. Antud maatriksis asendada iga element tema naabrite summaga.

Näiteks, $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 5 & 2 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 6 & 9 & 8 \\ 13 & 20 & 17 \\ 17 & 23 & 15 \\ 9 & 14 & 11 \end{pmatrix}$

3.40. Koostada antud maatriksi

(a) reasummade järjend;

(b) veerusummade järjend;

(c) pea- ja kõrvaldiagonaali summade (kahe-elemendiline) järjend.

3.41. Leida antud maatriksi tihedus, st nullist erinevate elementide esinemissagedus protsentides.

Näiteks maatriksi $\begin{pmatrix} 11 & 1 & 0 & 3 \\ 0 & 43 & 0 & 6 \\ 3 & 7 & 0 & 0 \end{pmatrix}$ tihedus on $58,3\%$ (nn hõredus siis $41,7\%$).

3.42. Antud ruutmaatriksis asendada nulliga need allpool peadiagonaali asetsevad elemendid, mis võrduvad vastava peadiagonaali suhtes sümmeetrilise elemendiga.

Näiteks, $\begin{pmatrix} 11 & 1 & 22 & 5 \\ 0 & 43 & 7 & 6 \\ 3 & 7 & 8 & 9 \\ 5 & 6 & 5 & 61 \end{pmatrix} \Rightarrow \begin{pmatrix} 11 & 1 & 22 & 5 \\ 0 & 43 & 7 & 6 \\ 3 & 0 & 8 & 9 \\ 0 & 0 & 5 & 61 \end{pmatrix}$.

3.43. Kontrollida, kas kaks antud maatriksit on võrdsed.

3.44. Antud maatriksis vahetada (kui võimalik) ridu selliselt, et peadiagonaalist ülevalpool oleksid ainult nullid.

Näiteks, $\begin{pmatrix} 0 & 5 & 3 & 0 \\ 1 & 0 & 0 & 0 \\ 7 & 0 & 8 & 4 \\ 2 & 0 & 0 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 5 & 3 & 0 \\ 7 & 0 & 8 & 4 \end{pmatrix}$.

3.45. Antud on maatriks A ja täisarv k.
Veerud maatriksis A nihutada tükliliselt $|k|$ kohta

- paremale, kui $k > 0$;
- vasakule, kui $k < 0$.

Näiteks, maatriksi

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \\ 26 & 27 & 28 & 29 & 30 \\ 31 & 32 & 33 & 34 & 35 \end{pmatrix}$$

veergude nihutamisel $k = 2$ kohta saadakse

$$A = \begin{pmatrix} 4 & 5 & 1 & 2 & 3 \\ 9 & 10 & 6 & 7 & 8 \\ 14 & 15 & 11 & 12 & 13 \\ 19 & 20 & 16 & 17 & 18 \\ 24 & 25 & 21 & 22 & 23 \\ 29 & 30 & 26 & 27 & 28 \\ 34 & 35 & 3 & 132 & 33 \end{pmatrix}.$$

3.46. Sorteerida elemendid antud $m \times n$ maatriksis a. Sorteeritud maatriksis on iga rida sorteeritud mittekasvavalt ja $a_{i-1n-1} \leq a_{i0}$, iga i korral, $i = 1 \ldots m-1$.

3.47. Antud bitimaatriksis leida rida, milles on kõige rohkem ühtesid.

3.48. Antud $m \times n$ maatriksis a vahetada pea- ja kõrvaldiagonaal. (Peadiagonaalil on i-s element $a_{i,i}$, kõrvaldiagonaalil aga $a_{i,n-1-i}$, $i = 0, 1, \ldots m-1$.)

Näiteks maatriksi

$$\begin{pmatrix} 11 & 1 & 0 & 3 \\ 0 & 43 & 0 & 6 \\ 3 & 7 & 0 & 0 \end{pmatrix}$$

pea- ja kõrvaldiagonaali vahetamisel saadakse maatriks

$$\begin{pmatrix} 3 & 1 & 0 & 11 \\ 0 & 0 & 43 & 6 \\ 3 & 0 & 7 & 0 \end{pmatrix}.$$

3.49. Antud ruutmaatriksis vahetada ülemine (ülalpool peadiagonaali asuv) kolmnurkmaatriks alumise kolmnurkmaatriksiga.

Näiteks, $\begin{pmatrix} 2 & 6 & 5 & 8 \\ 9 & 3 & 6 & 7 \\ 2 & 1 & 0 & 8 \\ 9 & 1 & 1 & 2 \end{pmatrix} \Rightarrow \begin{pmatrix} 2 & 9 & 2 & 9 \\ 6 & 3 & 1 & 1 \\ 5 & 6 & 0 & 1 \\ 8 & 7 & 8 & 2 \end{pmatrix}$.

3.50. Koondada antud maatriksis kõik veerud

(a) alla;

(b) üles.

Iga veerg koondatakse kui vastav järjend kas paremale (juhul a) või vasakule (juhul b). Järjendi koondamine: vt ülesanne 2.73.

Näiteks, maatriks

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 13 & 0 & 0 & 2 \\ 3 & 0 & 0 & 3 \\ 3 & 0 & 1 & 4 \\ 6 & 0 & 0 & 0 \end{pmatrix}$$

pärast üles koondamist :

$$\begin{pmatrix} 13 & 1 & 1 & 1 \\ 6 & 0 & 0 & 2 \\ 6 & 0 & 0 & 3 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

3.51. Loeme $m \times n$ maatriksi *rõhtvööks*

- selle maatriksi rea indeksiga $m/2$ (st keskmise rea), kui m on paaritu,
- selle maatriksi read indeksitega $\lfloor m/2 \rfloor$ ja $\lceil m/2 \rceil$, kui m on paarisarv;

püstvööks aga loeme

- selle maatriksi veeru indeksiga $n/2$ (keskmise veeru), kui n on paaritu,
- selle maatriksi veerud indeksitega $\lfloor n/2 \rfloor$ ja $\lceil n/2 \rceil$, kui n on paarisarv.

Arvutada antud maatriksi rõhtvöö elementide summa r ja püstvöö elementide summa p, ning juhul $r < p$ transponeerida see maatriks.

Näiteks, maatriksi

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 13 & 0 & 0 & 2 \\ 3 & 0 & 0 & 3 \\ 3 & 0 & 1 & 4 \\ 6 & 0 & 0 & 0 \end{pmatrix}$$

korral $r = 6$ ja $p = 2$ ning transponeerida ei tule.

3.52. Antud on maatriksid A ja B.
Konrollida, kas $B = A'$ (st kas B on transponeeritud A).

3.53. Antud bitimaatriksis leida (kui võimalik)

(a) suurim alammaatriks, mille igas nurgas asub 1;

(b) suurim ainult nullidest koosnev alammaatriks.

3.54. Koostada bitimaatriks, milles on antud arv (m) ridu ja antud arv (n) veerge ning iga ring selles maatriksis on kas ühtede ring või nullide ring, kusjuures ühtede ja nullide ringid paiknevad vaheldumisi.

(Siin: *ringi maatriksis* moodustavad alammaatriksi äärelahtrid, mis paiknevad maatriksi vastavatest äärtest ühel ja samal kaugusel.)

Näiteks,

$m = 4$ ja $n = 5$ korral koostakse (kaheringiline) maatriks $\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$,

$m = 7$ ja $n = 8$ korral aga neljaringiline maatiks $\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$.

3.55. Antud bitimaatriksis leida suurim ruut-alammaatriks,

(a) mille välimine ring (ääre-elementide hulk) koosneb ainult ühtedest;

(b) mis koosneb ainult ühtedest.

Näiteks, maatriksis $\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$ paikneb selliste alammaatriksite vasak ülemine nurk teise rea kolmandas veerus: 4×4 alammaatriks juhul a) ja 2×2 alammaatriks juhul b).

3.56. Antud on maatriks A ja täisarv k.
Iga elementide ring maatriksis a nihutada tükliliselt $|k|$ kohta

- päripäeva, kui $k > 0$;
- vastupäeva, kui $k < 0$.

Näiteks, maatriksi

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \\ 26 & 27 & 28 & 29 & 30 \\ 31 & 32 & 33 & 34 & 35 \end{pmatrix}$$

ringide nihutamisel $k = -3$ kohta saadakse

$$A = \begin{pmatrix} 4 & 5 & 10 & 15 & 20 \\ 3 & 14 & 19 & 24 & 25 \\ 2 & 9 & 13 & 29 & 30 \\ 1 & 8 & 18 & 28 & 35 \\ 6 & 7 & 23 & 27 & 34 \\ 11 & 12 & 17 & 22 & 33 \\ 16 & 21 & 26 & 31 & 32 \end{pmatrix}.$$

3.57. Antud bitimaatriksis leida nende ühtede arv, mis on ainsad oma reas ja oma veerus.

3.58. Kontrollida, kas antud maatriksis leidub rida, mille summa võrdub selle reaga sama numbriga veeru summaga. Näiteks, maatriksi

$$\begin{pmatrix} 3 & 1 & 2 & 11 \\ 0 & 0 & 3 & 6 \\ 5 & 0 & 17 & 0 \end{pmatrix}$$

korral on vastus jaatav, sest nii viimase (kolmanda) rea summa kui ka eelviimase (kolmanda) veeru summa on 22.

3.59. Antud $m \times n$ ($m, n \geq 3$) maatriksis leida suurima summaga 3×3 alammaatriks.

Näiteks, maatriksis

$$A = \begin{pmatrix} 9 & -1 & 8 & 4 & 2 & 7 & 3 \\ 5 & -1 & 9 & 2 & 3 & 0 & 3 \\ 0 & 3 & 0 & 5 & 8 & 8 & 2 \\ 0 & 1 & 1 & 7 & 8 & 3 & 2 \\ 2 & 0 & 5 & 9 & 6 & 7 & 4 \end{pmatrix}$$

on suurima summaga 3×3 alammaatriksiks $\begin{pmatrix} 5 & 8 & 8 \\ 7 & 8 & 3 \\ 9 & 6 & 7 \end{pmatrix}$ ülemise vasaku nurga indeksitega $(2,3)$ maatriksis A.

3.60. Antud naturaalarvu k ja $m \times n$ ($m, n \geq k$) maatriksi A korral koostada A kõikide $k \times k$ alammaatriksite summade maatriks.

Näiteks, $k = 3$,

$$A = \begin{pmatrix} 1 & 11 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 22 & 2 & 0 \\ 3 & 3 & 3 & 3 & 3 & 0 \\ 4 & 44 & 4 & 4 & 4 & 0 \\ 55 & 5 & 5 & 5 & 5 & 0 \end{pmatrix};$$

maatriksi A 3×3 alammaatriksite summad on $\begin{pmatrix} 28 & 48 & 38 & 33 \\ 67 & 87 & 47 & 38 \\ 126 & 76 & 36 & 24 \end{pmatrix}$.

3.61. Antud bitimaatriksi B korral leida (kui võimalik) selline indeks k, mille korral

- $B_{k,k} = 0$ või $B_{k,k} = 1$;
- reas indeksiga k on kõik elemendid (peale $B_{k,k}$) nullid;
- veerus indeksiga k on kõik elemendid (peale $B_{k,k}$) ühed.

Näiteks, maatriksi $\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$ korral indeks $k = 5$.

3.62. Antud bitimaatriksis

- iga rida, milles vähemalt ühe elemendi väärtus oli üks, muuta ühtede reaks;
- iga veerg, milles vähemalt ühe elemendi väärtus oli üks, muuta ühtede veeruks.

Näiteks, maatriksi $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$ korral on tulemuseks $\begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$.

3.63. Loetleda arvud, mis esinevad antud täisarvumaatriksi igas veerus.

Näiteks, maatriksi $\begin{pmatrix} 1 & 11 & 1 & 1 & 1 & 1 \\ 2 & 4 & 5 & 22 & 4 & 0 \\ 3 & 3 & 3 & 3 & 3 & 4 \\ 4 & 44 & 4 & 4 & 0 & 0 \\ 55 & 1 & 1 & 3 & 5 & 3 \end{pmatrix}$ igas veerus esinevad 1, 3 ja 4.

3.64. Kontrollida, kas antud maatriks on *Toeplitzi maatriks*.

3.65. Antud bitimaatriksis leida kõik teed kujul

```
0 0 ... 0
0
...
0 ,
```

kus alumine 0 (tee algus) asub maatriksi viimasel real ja parempoolne 0 (tee lõpp) asub maatriksi paremas ääres. Minimaalseks teeks oleks viimase rea lõpus asuv 0.

Näiteks, maatriksis

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

leidub 7 sellise kujuga teed:

$$\begin{pmatrix} & 0 & 0 & 0 & 0\rightarrow \\ & 0 & & & 0\rightarrow \\ 0 & 0 & 0 & 0 & 0\rightarrow \\ 0 & 0 & & & 0\rightarrow \\ \uparrow & \uparrow & & & \uparrow \end{pmatrix}.$$

3.66. Leida, mitu rida antud maatriksis on rangelt sorteeritud, st on kas rangelt kasvav või rangelt kahanev.

3.67. Kontrollida, kas antud bitimaatriks on sümmeetriline oma

(a) horisontaaltelje suhtes;

(b) vertikaaltelje suhtes.

3.68. Antud maatriksis paigutada elemendid ümber selliselt, et

- iga rida oleks sorteeritud mittekahanevalt;
- iga veerg oleks sorteeritud mittekahanevalt;
- mõlemad (ülalt-alla) diagonaalid oleksid sorteeritud mittekahanevalt.

Näiteks, $\begin{pmatrix} 0 & 2 & 1 & 7 \\ 4 & 5 & 6 & 0 \\ 3 & 0 & 2 & 8 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 0 & 0 & 1 \\ 2 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$.

3.69. Esitada tabelina arvujärjendi kõik (mittetühjad)

(a) alamjärjendid;

(b) osajärjendid.

Näiteks, järjendi $(1.0, -3.2, 6.63)$

a) mitteühjade alamjärjendite tabel: $\begin{pmatrix} 1.0 & & \\ -3,2 & & \\ 6.63 & & \\ 1.0 & -3.2 & \\ -3.2 & 6.63 & \\ 1.0 & -3.2 & 6.63 \end{pmatrix}$;

b) mitteühjade osajärjendite tabel: $\begin{pmatrix} 1.0 & & \\ -3,2 & & \\ 6.63 & & \\ 1.0 & -3.2 & \\ 1.0 & 6.63 & \\ -3.2 & 6.63 & \\ 1.0 & -3.2 & 6.63 \end{pmatrix}$.

3.70. Maatriksi *osanelikuks* nimetame maatriksi mingi rea või mingi veeru või peadiagonaali või kõrvaldiagonaali (paremast ülemisest nurgast) igat neljaelemendilist osajärjendit. Osanelikuid ei ole maatriksis, milles ridu ja veerge on alla nelja.

Leida antud arvumaatriksi kõige suurema (oma elementide) korrutisega osanelik.

Näiteks,

maatriksis

$$\begin{pmatrix} 7.4 & 6.5 & 2.9 & 3.2 & 0.3 & -0.9 & 0.6 \\ 4.0 & 6.0 & 2.5 & 6.2 & 0.4 & -8.6 & 7.0 \\ 0.4 & 5.7 & 1.9 & 5.2 & 8.5 & 7.1 & -0.4 \\ 7.1 & 3.5 & 3.4 & 5.0 & 1.5 & 4.2 & 7.2 \\ 4.7 & 7.5 & 3.4 & 7.8 & 7.2 & -6.4 & 2.4 \\ 2.6 & 3.8 & 3.6 & 0.0 & 3.1 & 8.8 & 8.6 \end{pmatrix}$$

on kõige suurema korrutisega (3438.9) osanelikuks eelviimase veeru osajärjend $(-8.6, 7.1, -6.4, 8.8)$,

maatriksis

$$\begin{pmatrix} 7.4 & 6.5 & 2.9 & 3.2 & 0.3 & 0.0 & 0.0 \\ 4.0 & 6.0 & 2.5 & 0.0 & 0.4 & -8.6 & 7.0 \\ 0.0 & 5.7 & 1.9 & 5.2 & 8.5 & 7.1 & -0.4 \\ 7.1 & 3.5 & 0.0 & 5.0 & 0.0 & 4.2 & 7.2 \\ 4.7 & 7.5 & 3.4 & 7.8 & 7.2 & 6.4 & 2.4 \\ 2.6 & 0.0 & 3.6 & 0.0 & 3.1 & 8.8 & 8.6 \end{pmatrix}$$

aga peadiagonaali osajärjend $(7.4, 6.0, 7.2, 8.8)$ (korrutisega 2813.184).

3.71. Leida antud maatriksi vähim element ja suurim element.

3.72. Leida ruutjuur antud naturaalarvulise maatriksi elementide summast, ümardatuna naturaalarvuks.

3.73. Pöörata antud ruutmaatriksis diagonaale üks kord

(a) päripäeva;

(b) vastupäeva.

Näide, diagonaalide päripäeva pööramise rakendamine kaks korda järjest:

$$\begin{pmatrix} 7 & 2 & -1 & 1 & 1 \\ 6 & 6 & 6 & 8 & 9 \\ 7 & 1 & 9 & 5 & 4 \\ 0 & 2 & 8 & 6 & 9 \\ 5 & 1 & 3 & 2 & 2 \end{pmatrix} \Rightarrow \begin{pmatrix} 5 & 2 & -1 & 1 & 7 \\ 6 & 2 & 6 & 6 & 9 \\ 7 & 1 & 9 & 5 & 4 \\ 0 & 6 & 8 & 8 & 9 \\ 2 & 1 & 3 & 2 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 2 & 2 & -1 & 1 & 5 \\ 6 & 6 & 6 & 2 & 9 \\ 7 & 1 & 9 & 5 & 4 \\ 0 & 8 & 8 & 6 & 9 \\ 1 & 1 & 3 & 2 & 7 \end{pmatrix}.$$

3.74. Antud on muutuva pikkusega ridadega tabel – järjendite järjend. Leida mittetühjades ridades esimesel kohal olevate arvude korrutis. (Null teguri korrutis loetakse võrdseks arvuga 1.)

Näiteks, tabeli

```
(
  (2,3,11),
  (),
  (5,4),
  (2,20),
)
```

korral on selliseks korrutiseks arv 20 (saadakse kui $2 \cdot 5 \cdot 2$).

3.75. Mõõtmetega $m \times n$ ($m,n > 4$) maatriksi *südamikuks* nimetame selle võimalikult keskel asuvat väiksemat (enamasti samades proportsioonides) alammaatriksit mõõtmetega $m' = f(m), n' = f(n)$, kus $f(x)$ on funktsioon südamiku mõõtme arvutamiseks sisaldava maatriksi vastatava mõõtme järgi.

Näide (kui $f(9) = 3$, $f(12) = 4$):

$$\begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
2 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
3 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
4 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
5 & 2 & 3 & \underline{4} & \underline{5} & \underline{6} & 7 & 8 & 9 \\
6 & 2 & 3 & \underline{4} & \underline{5} & \underline{6} & 7 & 8 & 9 \\
7 & 2 & 3 & \underline{4} & \underline{5} & \underline{6} & 7 & 8 & 9 \\
8 & 2 & 3 & \underline{4} & \underline{5} & \underline{6} & 7 & 8 & 9 \\
9 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
10 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
11 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
12 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9
\end{pmatrix}$$

– südamiku elemendid allakriipsutatud.

(a) Sätestada funktsiooni f arvutamise eeskiri ($f(x)$ väärtusteks võiksid olla näiteks $f(5) = 1$, $f(6) = 2$, $f(9) = 3$, $f(10) = 2$, $f(11) = 3$, $f(12) = 4$ vmt).

(b) Nullida antud maatriksi südamiku elemendid.

Südamiku nullimise näide:

$$\begin{pmatrix}
1 & 11 & 1 & 1 & 1 & 1 & 2 & 7 & 8 \\
2 & 4 & 5 & 22 & 4 & 9 & 3 & 7 & 8 \\
3 & 13 & 3 & \underline{33} & \underline{45} & \underline{4} & 3 & 7 & 8 \\
3 & 3 & 3 & \underline{31} & \underline{3} & \underline{2} & 3 & 7 & 8 \\
4 & 44 & 4 & 4 & 9 & 9 & 5 & 7 & 8 \\
55 & 1 & 1 & 3 & 5 & 3 & 4 & 7 & 8
\end{pmatrix} \Rightarrow \begin{pmatrix}
1 & 11 & 1 & 1 & 1 & 1 & 2 & 7 & 8 \\
2 & 4 & 5 & 22 & 4 & 9 & 3 & 7 & 8 \\
3 & 13 & 3 & 0 & 0 & 0 & 3 & 7 & 8 \\
3 & 3 & 3 & 0 & 0 & 0 & 3 & 7 & 8 \\
4 & 44 & 4 & 4 & 9 & 9 & 5 & 7 & 8 \\
55 & 1 & 1 & 3 & 5 & 3 & 4 & 7 & 8
\end{pmatrix}.$$

3.76. Antud on bittide ruutmaatriks ja selle ühe lahtri indeksid. Maatriksi antud lahtris asub sipelgas, kes võib mööda maatriksit liikuda järgmisel viisil. Kui lahtris, kus sipelgas asub, on bitt 0, siis muudab sipelgas selle bitiks 1, pöörab ennast vasakule ja liigub ühe sammu edasi naaberlahtrisse. Kui lahtris on 1, siis muudab sipelgas selle bitiks 0-ks, pöördub paremale ja liigub jälle sammu edasi. Kui sipelgas läheb oma sammuga maatriksis "üle ääre", siis ilmub ta vastasservas maatriksisse tagasi. Alguses on maatriksi kõik elemendid nullid.

Leida ühtede protsent maatriksis pärast seda, kui sipelgas on teinud etteantud arvu samme.

Tabel 1: Mõned linnadevahelised kaugused

	Elva	Haapsalu	Kuressaare	Narva	Pärnu	Rakvere	Tallinn	Tartu	Valga	Viljandi	Võru
Elva	0	267	308	211	156	152	216	27	60	70	75
Haapsalu	267	0	155	314	111	203	101	258	254	199	310
Kuressaare	308	155	0	429	152	313	216	330	295	249	351
Narva	211	314	429	0	299	116	212	184	271	265	252
Pärnu	156	111	152	299	0	183	129	178	143	97	199
Rakvere	152	203	315	116	183	0	99	126	212	151	193
Tallinn	216	101	216	212	129	99	0	189	252	161	257
Tartu	27	258	330	184	178	126	189	0	87	73	68
Valga	60	254	295	271	143	212	252	87	0	91	71
Viljandi	70	199	249	265	97	151	161	73	91	0	128
Võru	75	310	351	252	199	193	257	68	71	128	0

3.77. Tabeli 1 andmetel

(a) kontrollida, kas iga linna kaugus iseendast on 0;

(b) kontrollida, kas kaugus suvalisest linnas mingisse teise on sama, mis sealt tagasi;

(c) kontrollida, kas suvalisest linnast teise linna on otsetee alati lühem kui läbi mingi suvalise kolmanda linna;

(d) leida, millise kahe erineva Eesti linna vahel on vahemaa kõige väiksem;

(e) leida, millise kahe erineva Eesti linna vahel on vahemaa kõige suurem;

(f) leida linn, mille maksimaalne kaugus teistest on minimaalne;

(g) leida linn, millel on enda lähinaabruses (vähem kui 100 kilomeetrit) kõige rohkem naabreid;

(h) leida linn, millel on enda kaugnaabruses (rohkem kui 200km ja vähem kui 300 km) kõige rohkem naabreid;

(i) leida, millisest linnast millisesse saab minna läbi mingi kolmanda nii, et kogukilometraaž oleks vähim;

(j) teatava vesinikautoga saab tankimata sõita x kilomeetrit; sellele sõidukile sobivad tanklad asuvad vaid tabelis 1 toodud linnades; leida,

millise vähima tankimiskordade arvuga saab sõita ühest antud linnast teise antud linna;

(k) leida, milline peaks olema vesinikuauto tankimata sõidu vähim pikkus x, mille korral õnnestuks selle auto abil igast linnast saada igasse teise (vajadusel vahetankimistega);

(l) Eesti Kergejõustikuliit (EKJL) tahab korraldada ühes neist linnadest võistlused; sealjuures peab EKJL rahastama igast linnast võistluseks valitud linna sõitva ühe bussi sõidu võistlusele ja tagasi; leida, millist linna EKJL-le soovitada, kui eesmärk on vähima busside kogukilometraaži saavutamine.

4. Rekursiivne töötlemine

Käesolevas jaotises esitatakse rida ülesandeid, millest lähtudes sobib koostada rekursiivseid lahendusi nõudvaid programmeerimisülesandeid. Nendes tuleb muidugi sätestada mitmesuguseid erinõudeid. Näiteks: keelata meetodiväliste (staatiliste) muutujate kasutamine, nõuda kas unaarse või binaarse või muu asjakohase rekursiooniskeemi rakendamist, tarbetu ebaefektiivsuse vältimist jmt. Vt ka viimases, lisajaotises toodud rekursioonitehnika metoodilist juhendit.

4.1. Leida antud arvujärjendi elementide

(a) summa;

(b) miinimum.

4.2. Leida (kui võimalik) antud arvujärjendis (üks) arv, mis asub vahemikus $(0;1)$.

Näiteks, järjendis $(1.0, 0.7, 0.0, -0.9, 0.3)$ on selliseks arvuks kas 0.7 või 0.3.

4.3. Leida summa antud täisarvujärjendi elementidest, kus paarisarvulise väärtusega elemendid on võetud ühekordselt ja paarituarvulise väärtusega elemendid kahekordselt.

Näiteks, järjendi (2, 8, 3, 7, 10) korral on selliseks summaks 40.

4.4. Leida antud arvujärjendi positiivsete elementide kuupide summa.

Näiteks, järjendi $(2.0, 1.5, -18.0)$ positiivsete elementide kuupide summa on 11.375.

4.5. Leida antud arvujärjendi elementide kahekaupa korrutiste summa, kus esimene korrutatakse viimasega, teine eelviimasega, kolmas lõpust kolmandaga jne. Paaritu arvu liikmete korral keskmine element ainult liidetakse.

Näiteks, järjendi (1, 2, 3, 4, 5) korral on selliseks summaks 16 ($= 1 \cdot 5 + 2 \cdot 4 + 3$).

4.6. Leida selline summa antud arvujärjendi elementidest, kus esimene element võetakse ühekordselt, teine kahekordselt jne.

Näiteks, järjendi (2, 5, 0, 1, 3) korral on selliseks summaks 31 ($= 1 \cdot 2 + 2 \cdot 5 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 3$).

4.7. Leida antud arvujärjendi nullist erinevate elementide korrutis.

Näiteks, järjendi (0.0, 4.0, 2.7) nullist erinevate elementide korrutis on 10.8.

4.8. Antud arvujärjendis $a = (a_0, a_1, \ldots, a_n)$

(a) vahetada igas naabritepaaris paariliikmete väärtused (kui järjendis on paaritu arv elemente, siis viimase elemendi väärtus jääb muutmata);

(b) loendada erimärgiliste liikmetega naabritepaarid;

(c) vahetada igas naabritepaaris paariliikmete väärtused ja ühtlasi loendada erimärgiliste liikmetega naabritepaarid.

(Arvestada, et naabritepaari esimene liige on paarisarvulise indeksiga järjendis a.)

Näiteks,
(juht a) $(-5,-4,-1,4,-2,0,-8) \Rightarrow (-4,-5,4,-1,0,-2,-8)$;
(juht b) järjendis $(10,-10,-10,21,-21,3,3,7)$ on 3 erimärgiliste liikmetega paari.

4.9. Antud järjendi a elemendid jaotada kahte järjendisse b ning c, võttes vaheldumisi elemente järjendist a.

Näiteks, järjendist $a = (1,2,3,4,5)$ saadakse $b = (1,3,5)$ ja $c = (2,4)$.

4.10. Leida järjestikuste nullide arv antud täisarvujärjendi

(a) alguses;

(b) lõpus.

4.11. Antud on naturaalarv n. Loetleda kõik n-elemendilised bitivektorid.

4.12. Antud on naturaalarv n. Loetleda kõik n-elemendilised bitivektorid, mis algavad bitiga 0 ja ei sisalda naabrite paari 11.

Näiteks,

- $n = 1$ korral leidub ainult üks bitivektor

  0

- $n = 2$ korral – bitivektorid

  00
  01

- $n = 5$ korral – bitivektorid

  00000
  00001
  00010
  00100
  00101
  01000
  01001
  01010

4.13. Antud naturaalarvu n korral

(a) loendada,

(b) loetleda leksikograafilises järjestuses

n-elemendilised bitivektorid, mis algavad bitiga 1 ja ei sisalda naabrite paari 00.

4.14. Antud naturaalarvude n ja $k \leq n$ korral loetleda kõik pikkusega n bitivektorid, milles on k ühte.

4.15. Antud on naturaalarvud n ja $m(m < 10)$. Loetleda kõik n-kohalised arvud, mis on moodustatud numbritest $0,\ldots,m$.

Näiteks, n=2 ja m = 2 korral saadakse loetelu $00,01,02,10,11,12,20,21,22$.

4.16. Antud on naturaalarvude järjend a. Sellele vastav aritmeetiline avaldis saadakse kui a elementide rida, milles arvude vahele on pandud mingi binaarse tehte märk.

Näiteks mõned järjendist $a = (2,12,5,2)$ saadavad aritmeetilised avaldised:

```
2+12+5+2
2-12-5+2
2+12*5-2
2*12*5-2.
```

Loetleda kõik järjendist a saadavad aritmeetilised avaldised, mille väärused kuuluvad etteantud lõiku $[x;y]$, kui lubatud on binaarsed tehted

(a) $+$ ja $-$;

(b) $+$, $-$ ja $*$.

Näiteks, lõigu $[-15;15]$ ja ülaltoodud näitejärjendi korral leitakse (juhul b):

```
2+12-5+2 = 11
2+12-5-2 = 7
2+12-5*2 = 4
2-12+5+2 = -3
2-12+5-2 = -7
2-12+5*2 = 0
2-12-5+2 = -13
2*12-5*2 = 14
```

4.17. Antud naturaalarvude järjendi a ja arvu s korral

(a) loendada,

(b) loetleda

aritmeetilised avaldised, mille väärtuseks on s ja mis on esitatud järjendi a arvuga 0 eest täiendatud mingi osajärjendi arvude reana eraldajat $'+'$ või $'-'$ kasutades.

Näiteks, $a = (2,6,3,5)$ korral leidub

4 avaldist summaga $s = 6$:

$0+6$

$0+2+6+3-5$

$0-2+3+5$

$0-2+6-3+5$

ja 5 avaldist summaga $s = 0$:

0

$0+2+3-5$

$0+2+6-3-5$

$0-2-3+5$
$0-2-6+3+5$.

4.18. Antud n korral leida kõigi n-kohaliste arvude summa, mis algavad numbriga 3 ja mille iga järgnev number on kas 3 või 7.

Näiteks,
$n=2$ korral on selliste arvude (33 ja 37) summaks 70,
$n=5$ korral aga on selliste arvude summa 11377760.

4.19. Antud on naturaalarvud n ja k. Loendada võimalused esitada arv n k positiivse täisarvu summana, st leida erinevate järjendite $(p_1, p_2, \ldots, p_k)$ arv, kus $p_i > 0, i = 1,2,\ldots,k$ ja $p_1 + p_2 + \ldots + p_k = n$,

Näiteks,
$n=5$ ja $k=2$ korral on 4 võimalust (vastavad liidetavate järjendid on $(1,4)$, $(2,3)$, $(3,2)$ ja $(4,1)$);
$n=5$ ja $k=3$ korral on 6 võimalust (vastavad liidetavate järjendid on $(1,1,3)$, $(1,2,2)$, $(1,3,1)$, $(2,1,2)$, $(2,2,1)$ ja $(3,1,1)$).

4.20. Antud on naturaalarv n. Loetleda arvude (indeksite) $0,1,\ldots,n-1$ permutatsioonid

(a) kui kordused ei ole lubatud;

(b) kui kordused on lubatud.

Näiteks, $n=3$ korral on loeteluks (juhul a) arvude $0,1,2$ kordusteta permutatsioonid

```
0, 1, 2
0, 2, 1
1, 0, 2
1, 2, 0
2, 0, 1
2, 1, 0.
```

Kui $n=2$, siis (juhul b) arvude $0,1$ kordustega permutatsioonid on

```
0, 0
0, 1
1, 0
1, 1.
```

4.21. Koostada antud naturaalarvude järjendi permutatsioonide hulk.

Näiteks, järjendi $(1,2,3,2)$ permutatsioonide hulgaks on $\{(2,2,3,1)$, $(2,3,2,1)$, $(2,3,1,2)$, $(3,2,2,1)$, $(3,2,1,2)$, $(3,1,2,2)$, $(2,2,1,3)$, $(2,1,2,3)$, $(2,1,3,2)$, $(1,2,2,3)$, $(1,2,3,2)$, $(1,3,2,2)\}$.

4.22. Antud on täisarvud n ja k $(0 \le k \le n)$. Loetleda arvude (indeksite) $0,1,\ldots,n-1$ kombinatsioonid k kaupa

(a) kui kordused ei ole lubatud;

(b) kui kordused on lubatud.

Näiteks, $n = 5$, $k = 3$ korral on loeteluks (juhul a) arvude $0,1,2,3,4$ kordusteta kombinatsioonid kolme kaupa:

```
0, 1, 2
0, 1, 3
0, 1, 4
0, 2, 3
0, 2, 4
0, 3, 4
1, 2, 3
1, 2, 4
1, 3, 4
2, 3, 4.
```

Kui $n = 4$ ja $k = 3$, siis (juhul b) arvude $0,1,2,3$ kordustega kombinatsioonid kolme kaupa:

```
0, 0, 0
0, 0, 1
0, 0, 2
0, 0, 3
0, 1, 1
0, 1, 2
0, 1, 3
0, 2, 2
0, 2, 3
0, 3, 3
1, 1, 1
1, 1, 2
1, 1, 3
1, 2, 2
1, 2, 3
1, 3, 3
2, 2, 2
2, 2, 3
2, 3, 3
3, 3, 3.
```

4.23. Antud arvujärjendi ja arvu s korral

(a) leida (kui võimalik) üks summaga s osanelik (neljaelemendiline osajärjend);

(b) loetleda kõik summaga s osanelikud (neljaelemendilised osajärjendid).

Näiteks,
järjendi $(2,7,4,0,9,5,2,3)$ summaga 20 osanelikuteks on
$(2,7,9,2)$,
$(2,4,9,5)$,
$(7,4,0,9)$,
$(4,9,5,2)$.

4.24. Loetleda kõik hulgad, mida saab moodustada antud järjendi elementidest.

Näiteks, järjendi $(2,1,2,3,1,2,0,0,3)$ elementidest saab moodustada järgmised hulgad:
$\{\},\{0\},\{0,1\},\{1\},\{0,2\},\{2\},\{0,1,2\},\{0,3\},\{1,2\}$,
$\{3\},\{0,1,3\},\{1,3\},\{0,2,3\},\{2,3\},\{0,1,2,3\},\{1,2,3\}$.

4.25. Loetleda antud järjendi kõik erinevad osajärjendid (st leida osajärjendite hulk).

Näiteks, järjendi $(1,1,1,3)$ erinevad osajärjendid on
(1, 1, 1, 3), (1, 1, 3), (1, 3), (3), (1, 1, 1), (1, 1), (1), ().

4.26. Antud on unikaalsetest elementidest koosnev järjend a. Leida järjendi a elementide kõikvõimalikud jaotused kahte rühma (kahte järjendisse), kusjuures kummagi rühma elemendid säilitavad oma järjendis a oleva järjestuse.

Näiteks, järjendi $a = (1,2,3)$ jaotused kahte rühma:

$$(1,2,3)()$$
$$(1,2)(3)$$
$$(1,3)(2)$$
$$(1)(2,3)$$
$$(2,3)(1)$$
$$(2)(1,3)$$
$$(3)(1,2)$$
$$()(1,2,3)$$

4.27. Leida antud järjendi jaotused etteantud arvuks rühmadeks – mittetühjadeks järjestikusteks alamjärjenditeks.

Näiteks, järjendi $(1,2,3,4,5)$ jaotused kolmeks rühmaks:

$$(1)(2)(3,4,5)$$
$$(1)(2,3)(4,5)$$
$$(1)(2,3,4)(5)$$
$$(1,2)(3)(4,5)$$
$$(1,2)(3,4)(5)$$
$$(1,2,3)(4)(5)$$

4.28. Leida antud arvujärjendi jaotus mittetühjadeks mittekahanevateks järjestikusteks alamjärjenditeks.

Näiteks,
arvujärjend $(24,28,44,35,40,27,31,31,46,29,52,20,30)$ jaotub mittekahanevateks alamjärjenditeks $(24,28,44)$, $(35,40)$, $(27,31,31,46)$, $(29,52)$ ja $(20,30)$.

4.29. Leida antud järjendi kõikvõimalikud jaotused rühmadesse – mittetühjadeks järjestikusteks alamjärjenditeks.

Näiteks, järjendi $(11,12,13,14)$ kõikvõimalikud jaotused:
$(11,12,13,14)$
$(11)(12,13,14)$
$(11,12)(13,14)$
$(11,12,13)(14)$
$(11)(12)(13,14)$
$(11)(12,13)(14)$
$(11,12)(13)(14,)$
$(11)(12)(13)(14)$.

4.30. Leida antud arvujärjendi kõik teisendid, mis saadakse selle mõnede nullist erinevate elementide asendamisel vastandarvudega. Üheks teisendiks lugeda ka antud järjend ise.

Näiteks,
järjendi $(1,2,0)$ teisenditeks on $(1,2,0)$, $(1,-2,0)$, $(-1,2,0)$ ja $(-1,-2,0)$.

4.31. Loetleda (nummerdatult) antud naturaalarvu n kõik lahutused liidetavateks 3 ja 5.

Näiteks, arvu 18 liidetavateks saavad olla
1) 333333
2) 3555
3) 5355
4) 5535
5) 5553

4.32. Antud on naturaalarvud m ja x. Loendada võimalused esitada arv m summana positiivsetest liidetavatest, mis pole suuremad kui x ja kus liidetavad on suuruse järjestuses. Teisisõnu, tuleb leida erinevate mittekahanevalt sorteeritud järjendite arv, mille elemendid on poollõigust (0; x] ja nende elementide summa on m.

Näiteks,
$m = 4$ ja $x = 2$ korral on vastuseks 3 (vastavad liidetavate järjendid on
$(1,1,1,1)$, $(1,1,2)$ ja $(2,2)$);
$m = 7$ ja $x = 3$ korral on vastuseks 8 (vastavad liidetavate järjendid on
$(1,1,1,1,1,1,1)$, $(1,1,1,1,1,2)$, $(1,1,1,2,2)$, $(1,2,2,2)$, $(1,1,1,1,3)$, $(1,1,2,3)$, $(2,2,3)$ ja $(1,3,3)$).

4.33. Antud on naturaalarvud n ja k. Loendada järjendid, mille elemendid on hulgast $\{1,2\}$, elementide summa on n ja ei 1 ega 2 esine üle k korra.

Näiteks,

$n=4$ ja $k=2$ korral on vastuseks 4 (vastavad liidetavate järjendid on $(1,1,2)$, $(1,2,1)$, $(2,1,1)$, $(2,2)$);

$n=5$ ja $k=2$ korral on vastuseks 3 (vastavad esitused on $(1,2,2)$, $(2,1,2)$, $(2,2,1)$).

4.34. Järjendina on antud teatud kaupade hinnad eurodes. Loendada kaupade sellised valikud (igast kaubast mitte rohkem kui üks eksemplar), mille korral valitud kaupade koguhind ja valimata jäänud kaupade koguhind ei erine teineteisest rohkem kui 2 euro võrra.

Näiteks, hindade $(1,2,3,4,5)$ korral on selliseid valikuid 6 (vastavad komplektid on $(1,2,4)$, $(1,2,5)$, $(1,3,4)$, $(2,5)$, $(3,4)$ ja $(3,5)$).

4.35. Antud on naturaalarv s. Loetleda kõik sellised naturaalarvulised mittekahanevalt sorteeritud järjendid, mille elementide summa on s.

Näiteks, summa $s=4$ annavad järjendid $(1,1,1,1)$, $(1,1,2)$, $(1,3)$, $(2,2)$, (4).

4.36. Antud on naturaalarvud m ja x.

(a) Loendada võimalused esitada arv m summana erinevatest liidetavatest, mis ei ole väiksemad kui x, ja kus liidetavad on järjestatud kasvavalt.

(b) Loetleda kõik sellised naturaalarvulised, kasvavalt sorteeritud järjendid, mille elemendid on paarikaupa erinevad, ei ole väiksemad kui x ning mille summa on m.

Näiteks,

$m=3$ ja $x=1$ korral on tulemuseks järjendid $(1,2)$ ja (3),

$m=9$ ja $x=2$ korral $(2,3,4)$, $(2,7)$, $(3,6)$, $(4,5)$ ja (9).

4.37. Loetleda antud järjendi kõik mittetühjad alamjärjendid.

Näiteks,

järjendi $(1,-2,3)$ alamjärjenditeks on (1), $(1,-2)$, $(1,-2,3)$, (-2), $(-2,3)$, (3).

4.38. Antud järjendi korral

(a) loetleda,

(b) loendada

kõik selle mittetühjad osajärjendid.

Näiteks, järjendi $(1,-2,3)$ osajärjendite loeteluks on (1), $(1,-2)$, $(1,-2,3)$, $(1,3)$, (-2), $(-2,3)$, (3) ja loendamise tulemuseks 7.

4.39. Loendada võimalused valida antud järjendist 0 või enam elementi nii, et kui valitud elementidest moodustada uus järjend elementide järjestust muutmata, siis uue järjendi ükski kaks järjestikust elementi poleks võrdsed.

Näiteks, järjendi $(1,2,3)$ korral on 8 võimalust (sobivate valikute indeksite järjendid on $(0,1,2),(0,1),(0,2),(0),(1,2),(1),(2),()$); järjendi $(2,0,2,2)$ korral on 10 võimalust (sobivate valikute indeksite järjendid on $(0,1,2),(0,1,3),(0,1),(0)$, $(1,2),(1,3),(1),(2),(3),()$).

4.40. Olgu antud järjend $a=(a_1,a_2,\ldots,a_n)$ ja arv k, $0<k\leq n$. Leida järjendi a elementide selliste k-kaupa kombinatsioonide hulk, kus kombinatsiooni esimeseks elemendiks on a_1.

Näiteks järjendi $a=(0,1,1,2,3,5)$ ja $k=3$ korral on selliste kombinatsioonide hulgaks:
$\{(0,1,1),(0,1,2),(0,1,3),(0,1,5),(0,1,2),(0,1,3),(0,1,5),(0,2,3),(0,2,5),(0,3,5)\}$.

4.41. Antud on muutuva pikkusega (mittetühjade) ridadega tabel – järjendite järjend. Loendada võimalused valida antud tabelist 2 või enam rida nii, et igast kahest järjestikku valitud reast vasakpoolse viimane element võrdub parempoolse esimese elemendiga.

Näiteks, tabeli

1. $(2,3)$,
2. $(3,4,5)$,
3. $(3,6)$,
4. (5)

korral on võimaluste arvuks 4. Vastavad valikud on read numbritega

1. 2. 4.
1. 2.
1. 3.
2. 4.

4.42. Leida antud järjendi $a=(a_1,a_2,\ldots,a_n)$ elementide paarituarvulise k kaupa kombinatsioonide hulk, $k=1,3,\ldots m$, kus $m=\begin{cases} n, \text{ kui } n \text{ on paaritu,} \\ n-1, \text{ kui } n \text{ on paarisarv.}\end{cases}$

Näiteks järjendi $(1,3,6,10)$ korral on selliste kombinatsioonide hulgaks (1- ja 3-kaupa): $\{(1),(3),(6),(10),(1,3,6),(1,3,10),(1,6,10),(3,6,10)\}$.

4.43. Loendada antud naturaalarvu esitused positiivsete täisarvude summana, mille järjestikused liidetavad erinevad täpselt 1 võrra.

4.44. Loetleda antud naturaalarvu kõik esitused liidetavate 3 ja 4 summana, kus kumbki liidetav esineb vähemalt ühe korra. Liidetavate järjestuse poolest erinevad esitused lugeda erinevaks, järjestuse poolest sama esitus korduda ei või.

4.45. Loendada antud järjendi kõigi kuni k positiivsest elemendist koosnevad komplektid, kui järjestikustelt kohtadelt elemente komplektidesse ei valita.

Näiteks, järjendi $(1,2,-3,4,-5,6)$ korral on vastuseks 11; võimalikud komplektid: $(1),(2),(4),(6),(1,4),(1,6),(2,4),(2,6),(4,6),(1,4,6),(2,4,6)$.

4.46. Loetleda antud järjendi elementide kõikvõimalikud sellised jaotused kahte rühma, kus võrdsed elemendid on paigutatud koos samasse rühma.

Näiteks, järjendi $(5,1,0,1,5,1)$ jaotused on
$(0,1,1,1,5,5)$ ja $()$
$(0,1,1,1)$ ja $(5,5)$
$(0,5,5)$ ja $(1,1,1)$
(0) ja $(1,1,1,5,5)$
$(1,1,1,5,5)$ ja (0)
$(1,1,1)$ ja $(0,5,5)$
$(5,5)$ ja $(0,1,1,1)$
$()$ ja $(0,1,1,1,5,5)$.

4.47. Kontrollida, kas järjendis leidub kaks ühisosata ja sama summaga osajärjendit.

Näiteks, järjendis $(1,2,4,5,6,20)$ leidub, sest $4+5=1+2+6=9$, kuid järjendis $(1,3,7)$ mitte.

4.48. Loetleda antud pikkusega n bitijärjendid, mille esimeses ja teises pooles on sama palju ühtesid. Paarituarvulise n korral keskmist bitti arvesse ei võeta.

Näiteks, $n=5$ korral saadakse loetelu
01001 11011 10001 01010 00000 10110 01101 11111 10101 01110 00100 10110.

4.49. Antud on n-elemendiline järjend a ja naturaalarv k, $k \leq n$.

Leida järjendi a selline jaotus k rühmaks, et järjestikuste rühmade summade vahede ruutkeskmine oleks minimaalne.

Näiteks, järjendi $a=(1.1,0.9,3.8,2.7,0.1,1.5,2.6,0.3)$ ja $k=4$ korral oleks selliseks neljaksjaotuseks $((1.1,0.9)(3.8)(2.7,0.1,1.5)(2.6,0.3))$ summade vahede ruutkeskmisega 1.348,

$$1.348 = \sqrt{\frac{(2.0-3.8)^2+(3.8-4.3)^2+(4.3-2.9)^2}{3}}\,.$$

4.50. Loetleda antud naturaalarvu $n>7$ kõik esitused liidetavate 2 ja 3 summana. Liidetavate järjekorra poolest erinevad esitused lugeda erinevateks.

Näiteks, arvu $n=8$ liidetavate komplektideks on $(3,3,2)$, $(3,2,3)$, $(2,3,3)$ ja $(2,2,2,2)$.

4.51. Loetleda antud naturaalarvu $n>1$ kõik esitused liidetavate 1 ja 2 summana. Liidetavate järjekorra poolest erinevad esitused lugeda samaks.

Näiteks, arvu $n=3$ liidetavate komplektideks on $(1,1,1)$ ja $(1,2)$.

4.52. Antud on naturaalarvud n ja k. Loendada võimalused esitada arv n liidetavate 1 ja 2 summana, kus kumbki liidetav ei esine üle k korra. Esitused, mis erinevad liidetavate järjestuse poolest, lugeda erinevateks.

Näiteks,
$n=4$ ja $k=2$ korral on 4 võimalust (liidetavate komplektid: $(1,1,2)$, $(1,2,1)$, $(2,1,1)$, $(2,2)$);

$n = 5$ ja $k = 2$ korral on 3 võimalust (liidetavate komplektid: $(1,2,2)$, $(2,1,2)$, $(2,2,1)$).

4.53. Loetleda antud naturaalarvu n kõik esitused liidetavate 3, 4 ja 5 summana.

Näiteks, arvu $n = 9$ liidetavate komplektideks on $(5,4)$, $(4,5)$ ja $(3,3,3)$.

4.54. Loetleda antud naturaalarvu n kõik esitused liidetavate 2, 4 ja 6 summana. Esituses ei tohi olla kahte ühesugust liidetavat järjest. Liidetavate järjekorra poolest erinevad esitused loetakse erinevateks.

Näiteks, arvu $n = 8$ liidetavate komplektideks on $(2,4,2)$, $(2,6)$, $(6,2)$ (võimalused $(2,2,2,2)$, $(2,2,4)$, $(4,2,2)$ ja $(4,4)$ pole lubatud).

4.55. Antud on erinevate raamatute hindade järjend (võib sisaldada korduvaid hindu).

(a) Kontrollida, kas on võimalik valida raamatute komplekt, mille koguhind kuulub lõiku $[50;55]$ eurot.

(b) Leida (kui võimalik) sellise vähima raamatukomplekti koguhind, mis kuulub lõiku $[50;70]$ eurot.

Igat raamatut (hinda järjendist) võib komplekti võtta maksimaalselt üks kord.

4.56. Antud järjendis on toodud $n \geq 19$ erineva toote hinnad.

Leida toodete sellise valiku maksimaalne koguhind, mis ei ületa 100 eurot ja kus igat toodet (hinda järjendist) on võetud maksimaalselt

(a) üks kord;

(b) kaks korda.

4.57. Antud järjendis on toodud 25 erineva toote hinnad.

Leida vähim toodete sellise valiku koguhind, mis ületab 100 eurot ja kus igat toodet on võetud maksimaalselt 2 korda. Eeldada, et toodete koguhind ületab 100 eurot.

4.58. Antud järjendis on toodud $n \geq 32$ erineva toote hinnad.

Leida (kui võimalik) lühima sellise valiku pikkus, mille koguhind jääb lõiku [100; 110] ja kuhu ühtegi toodet ei ole valitud rohkem kui 1 kord.

4.59. Antud on $n > 15$ inimese kaale sisaldav täisarvujärjend.

Leida inimeste kaalude selline kaheks osaks jaotus, kus osade kaalude summad on võimalikult lähedased.

4.60. Jukul on taskus rahamündid vääringutega $(a_1, a_2, \ldots, a_n)$, igas vääringus münti täpselt 2 eksemplari. Ta soovib poes teha ostu koguhinnaga s.

Leida võimalikult väike väärtus $v \geq s$, mida on võimalik nende müntidega maksta (st leida, kui täpselt saab Juku ostu eest maksta).

Eeldada, et Jukul olevast rahast piisab ostu sooritamiseks.

4.61. Jukul on kodus rahamündid vääringutega $(a_1, a_2, \ldots, a_n)$, igas vääringus münte vähemalt 10 euro väärtuses. Ta soovib poes teha ostu koguhinnaga s.

Leida (kui võimalik) minimaalne müntide arv, millega Juku saaks poes maksta täpselt summa s eurot $(s < 10)$.

4.62. Leida vähim euromüntide arv, mille koguväärtuseks on antud rahasumma.

4.63. Antud on erinevate raamatute hindade järjend (võib sisaldada korduvaid hindu).

Leida kõikide selliste raamatukomplektide arv, mille koguhind jääb lõiku $[50;100]$ eurot, kui

(a) igat raamatut võib valikusse võtta maksimaalselt üks kord;

(b) iga raamatu peab valikusse võtma üks või kaks korda.

4.64. Antud on erinevate raamatute hindade järjend (võib sisaldada korduvaid hindu).

Leida kõikide selliste raamatukomplektide arv, mille koguhind ei erine 100 eurost üle 50 sendi, kui igat raamatut võib valikusse võtta maksimaalselt üks kord.

4.65. Antud järjendis on toodud poes müügil olevate 17 eri sorti šokolaadi(tahvlite) hinnad. Juku tahab osta komplekti šokolaade.

Loendada kõikvõimalike selliste komplektide koguhinnad, mis ületavad 25 eurot, kui Juku võtaks igat šokolaadi

(a) kuni 2 eksemplari (mõne võib ka võtmata jätta);

(b) kas 1 või 2 eksemplari.

4.66. Antud on parteide nimetuste järjend ja parteide mandaatide (arvude) järjend.

Loetleda võimalused nendest parteidest enamusvalitsuse moodustamiseks, st loetleda parteide sellised kombinatsioonid, mille korral nende mandaatide summa on vähemalt 51.

4.67.

Antud on naturaalarvud m ja n. Loetleda kõik järjendid pikkusega $m+n$, mis koosnevad kasvavalt paiknevatest arvudest $1, 2, \ldots, m$ ja kahanevalt paiknevatest arvudest $-1, -2, \ldots, -n$.

Näiteks, järjendite loetelu $m = 3$ ja $n = 2$ korral:
$(1,2,3,-1,-2)$
$(1,2,-1,3,-2)$
$(1,-1,2,3,-2)$
$(-1,1,2,3,-2)$
$(1,2,-1,-2,3)$
$(1,-1,2,-2,3)$
$(-1,1,2,-2,3)$

$(1,-1,-2,2,3)$
$(-1,1,-2,2,3)$
$(-1,-2,1,2,3)$.

4.68. Antud on kümnendnumber n ja arvu $x > 0$ kümnendnumbrite järjend a. Leida arvu $x+n$ kümnendnumbrite järjend b.

Näiteks,
kui $a = (2,5,1)$ ja $n = 5$, siis $b = (2,5,6)$;
kui $a = (0,2,5,1)$ ja $n = 5$, siis $b = (0,2,5,6)$;
kui $a = (9,9,7)$ ja $n = 8$, siis $b = (1,0,0,5)$;
kui $a = (4,5,9,9,7)$ ja $n = 5$, siis $b = (4,6,0,0,3)$.

4.69. Antud on $n > 18$ eseme kaale sisaldav järjend.

Loetleda kõikide selliste esemekomplektide kogukaalud, milles on 12, 13 või 14 eset.

4.70. Antud on n eseme kaale sisaldav järjend ja seljakoti kandevõime m (lubatud suurim kaal).

Leida võimalikult suure kogukaaluga k esemete valik, nii et $k \leq m$.

4.71. Arvutada 4×8 maatriksi A elementideks
A_{ij} = Ackermanni funktsiooni väärtus argumentidel i, j ($i = 0 \ldots 3$, $j = 0 \ldots 7$).

4.72. Antud on naturaalarv $n \leq 10$. Loetleda (nummerdatult) kõik n-kohalised kümnendarvud, milles numbrid on

(a) kasvavas,

(b) kahanevas

järjestuses.

Näiteks, $n = 9$ korral (juhul b) saadakse loetelu

```
1) 987654321
2) 987654320
3) 987654310
4) 987654210
5) 987653210
6) 987643210
7) 987543210
8) 986543210
9) 976543210
10)876543210
```

4.73. Loetleda antud järjendi osanelikud (neljaelemendilised osajärjendid), mille kahe keskmise elemendi aritmeetiline keskmine võrdub neliku aritmeetilise keskmisega.

Näiteks, järjendis $(-2,5,-1,4,2,9,7,3,-2)$ leidub kuus sellist osanelikut:
$(-2,5,2,9)$, $(-2,-1,2,3)$, $(5,-1,4,-2)$, $(5,-1,9,3)$, $(-1,4,2,7)$ ja $(4,2,9,7)$.

4.74. Antud on naturaalarvud n ja s. Loetleda kõik n-kohalised arvud, mille ristsumma on s.

Näiteks, kolmekohalised arvud ristsummaga 23:
$599, 689, 698, 779, 788, 797, 869, 878, 887, 896, 959, 968, 977, 986, 995$.

4.75. Leida, mitmel erineval viisil (sammude erineva reaga) saab jõuda trepi n-ndale astmele, kui ühe sammuga võtta

(a) kas 1 või 2 astet;

(b) kas 1, 2 või 3 astet.

4.76. Antud on mingi positiivsete arvude hulk ning arv $k > 1$.

Leida (kui võimalik) selle hulga klassijaotus k alamhulgaks, kus alamhulgad on ühe ja sama summaga.

Näiteks, hulk $\{3, 12, 14, 1, 22, 2, 24\}$
jaotub kolmeks ($k = 3$) alamhulgaks $\{2, 24\}$, $\{3, 1, 22\}$ ja $\{12, 14\}$ summadega 26, aga ka nt kaheks ($k = 2$) alamhulgaks $\{12, 1, 2, 24\}$ ja $\{3, 14, 22\}$ summadega 39, kuid ei jaotu neljaks ($k = 4$) võrdsete summadega alamhulgaks.

5. Sõnede töötlemine

I Iteratiivselt

Sõne on käsitletav sümbolite järjendina. Osajärjendi ja alamjärjendi mõistetele vastavad siin *osasõne* ja *alamsõne*. Sõnet, milles sümboliteks on tähed (mingist tähestikust), nimetame ka sõnaks.

Sümbol on sõnes *unikaalne*, kui esineb sõnes parajasti üks kord. *Heterogramm* on sõne, milles kõik sümbolid on unikaalsed. Kaks sõna on teineteise *anagrammid*, kui nad langevad kokku või erinevad üksnes sümbolite järjestuse poolest.

5.1. Leida sõnekujul antud naturaalarvu kõigi kümnendnumbrite summa (ehk ristsumma) sõnena.
Näiteks, sõnekujulise arvu *"23829"* ristsumma sõnena on *"24"*.

5.2. Transponeerida antud sõne.
Näiteks, *"oinas"* $\Rightarrow$ *"sanio"*.

5.3. Leida antud sümboli korduste arv antud sõnes.

5.4. Loetleda antud sõne kõik pikkusega 3 alamsõned.
Näiteks, sõne *"oinas"* pikkusega 3 alamsõned on *"oin"*, *"ina"* ja *"nas"*.

5.5. Kontrollida, kas antud sõne on heterogramm.

5.6. Loetleda antud sõnes unikaalsed sümbolid.

5.7. Leida antud sõnes esinevate (erinevate) sümbolite hulk.
Näiteks, sõne *"Vanaemake"* sümbolite hulgaks on { 'V', 'a', 'n', 'e', 'm', 'k'}.

5.8. Antud on sõna s pikkusega n. Koostada järjend $a = (a_1, a_2 \ldots, a_n)$, kus a_i $(i = 1, 2, \ldots, n)$ on sõna i-ndast sümbolist (koodi poolest) väiksemate või võrdsete sümbolite arv sellele sümbolile sõnas s järgnevate sümbolite seas.
Näiteks, kui $s =$*"oinas"*, siis $a = (3,1,1,0,0)$; kui $s =$*"lammas"*, siis $a = (2,1,2,1,0,0)$.

5.9. Leida (kui võimalik) kahe sõne pikim ühisprefiks.
(Sõne *prefiksiteks* on selle alguses paiknevad alamsõned.)

5.10. Antud sõnest eemaldada

(a) iga sümbol, mida esineb nii sellest eespool kui ka tagapool;

näiteks *"cacdacea"* $\Rightarrow$ *"cadcea"*;

(b) sümbolite kordumised, jättes alles vaid esimesena esinenu; näiteks *"vanapagan"* $\Rightarrow$ *"vanpg"* ning *"Java keel on tore"* $\Rightarrow$ *"Jav kelontr"*;

(c) need sümbolid, millele üleeelnev või ülejärgmine sümbol on samasugune; näiteks *"vanaema"* $\Rightarrow$ *"vnema"* ning *"bibigon"* $\Rightarrow$ *"gon"*;

(d) kõik alamsõne “*aj*” esinemised, mis ei kuulu alamsõne “*aaj*” koosseisu; näiteks “*maja*” ⇒ “*ma*”, kuid “*maaja*” ⇒ “*maaja*”.

5.11. Sorteerida sümbolid antud sõnes.
Näiteks, “*bacdae*” ⇒ “*aabcde*”.

5.12. Kontrollida, kas üks antud sõne on teise antud sõne anagramm.

5.13. Loetleda antud sõnes esinevad *kordusgrupid* – võimalikult pikad ühesugustest sümbolitest koosnevad, vähemalt pikkusega 2 alamsõned.
Näiteks, sõnes “*kuuuurijasse kuulub*” leidub kolm kordusgruppi: “*uuuu*”, “*ss*” ja “*uu*”.

5.14. Leida antud sõnes esinevate erinevate sümbolite arv.
Näiteks, sõnes “*Jaanilinn*” on 5 erinevat sümbolit.

5.15. Antud on kaks sõnet. Loetleda need sümbolid, mida esineb vaid ühes neist, ja sealgi 1 kord.
Näiteks, sõnedes “*kaalikas*” ja “*kalle*” on sellisteks sümbolid ’*i*’, ’*s*’ ning ’*e*’.

5.16. Loetleda kahe antud sõne need sümbolid, mida esineb täpselt 1 kord mõlemas sõnes.
Näiteks, sõnede “*Tore oli*” ja “*oi tere*” korral ’ ’, ’*i*’ ja ’*r*’.

5.17. Loetleda antud sõne kõik vähemalt 2-sümbolilised alamsõned, mida selles sõnes esineb

(a) vähemalt 2 korda;

(b) vähemalt 3 korda.

Näiteks,
(juht a) sõne “*terekesttеretere*” korral “*te*”, “*ter*”, “*tere*”, “*ere*”, “*er*”, “*re*”;
(juht b) sõne “*terekesttereter*” korral “*te*”, “*ter*”, “*er*”.

5.18. Eemaldada antud sõnest need sümbolid, mida esineb selles sõnes ka eespool.
Näiteks, “*terekesttereter*” ⇒ “*terks*”.

5.19. Antud on heterogramm s ja selle üks osasõne s'. Koostada sõne s'', mis on saadud sõnest s sõne s' sümbolite eemaldamisel.
Näiteks, s =“*abcdefg*”, s' =“*bdeg*” ⇒ s'' =“*acf*”.

5.20. Antud on lause, milles kõik sõnad peale esimese algavad väiketähega; esimene sõna ei ole pärisnimi. Muuta esimese sõna esitäht väiketäheks. Seejärel sorteerida leksikograafiliselt sõnad lauses, jättes kirjavahemärgid oma kohale. Lõpuks muuta esikohale saanud sõna esitäht suurtäheks.
Näiteks, “*Tere, mina olen siin!!*” ⇒ “*Mina, olen siin tere!!*”

5.21. Kontrollida, kas üks antud sõne on teisest antud sõnest saadav mõnede sümbolite vaheltkustutamise teel. (Täpsemalt: kas üks antud sõne on teise antud sõne osasõne.)

Näiteks, sõne "*kurat*" on saadav sõnest "*kuk.u.pr–a**t*" (on viimase osasõne).

5.22. Antud (mitteliit)sõnas märkida sidekriipsuga võimalikud poolituskohad vastavalt reeglitele:

- poolitada võib enne kaashäälikut, millele järgneb täishäälik;
- enne poolituskohta peab sõna algusesse jääma vähemalt kaks tähte;
- pärast poolituskohta peab sõna lõppu jääma vähemalt kaks tähte.

Näiteks, "*vastavalt*" $\Rightarrow$ "*vas-ta-valt*".

5.23. Leida sõne kujul antud aritmeetilise avaldise väärtus, kui operandideks on naturaalarvud ja teheteks

(a) ainult liitmised (tehtemärk $+$);

(b) liitmised, lahutamised, korrutamised (tehtemärk $+$, $-$ või $*$).

5.24. Antud teksti korral leida pikim sõna ja sõnade keskmine pikkus ning koostada tabel, kus on näidatud iga pikkusega sõnade protsent.

5.25. Loendada antud sõnes kõik kohad, kus esineb järjest kaks ühesugust eesti vokaali (kumbki võib olla nii väike- kui ka suurtähena).

Näiteks sõnedes "*Aasta uus saba suus*" ja "*Kuuuurija*" on kummaski 3, sõnes "*Linnuke laulis tilulõõ-tilulüü*" aga 2 sellist kohta.

5.26. Leida kaashäälikuühendite arv antud tekstis. Kaashäälikuühendiks on tekstis kõrvuti olevatest kaashäälikutest koosnev (võimalikult pikk) alamsõne.

5.27. Antud on tekst, kus sõnad on eraldatud vähemalt ühe tühikuga. Loetleda sõnad, milles mõni täht esineb kolm või enam korda järjest.

Näiteks, tekstis "*Ansambli Jääään igaaastane masssalvestus Kuuuurija saates.*" on 4 sellist sõna: "*Jääään*", "*igaaastane*", "*masssalvestus*" ja "*Kuuuurija*".

5.28. Koostada sõne, mille saab antud sõnest tema kõigi (tõstutundetult) ühesugustest sümbolitest koosnevate (pikimate) lõikude asendamisel vastavate lõikude esimese sümboliga.

Näiteks,
"*Halloo kosmos!*" $\Rightarrow$ "*Halo kosmos!*";
"*Jaava pähhhh!*" $\Rightarrow$ "*Java päh!*";
"*Aaaaaaarghhh!!!!!!*" $\Rightarrow$ "*Argh!*".

5.29. Antud tekstist (sõnad ja tühikud) moodustada lühendatud tekst: iga vähemalt kahest tähest koosnev sõna panustab lühendisse kaks esimest tähte, millest esimene on suur, teine väike; ühetäheline sõna panustab oma ainsa tähe ja see on suur; kõigi sõnade lühendid kirjutatakse järjest ilma vahedeta.

Näiteks,
"*Keskerakond Reformierakond*" ⇒ "*KeRe*";
"*Prawo i sprawiedliwoúã*" ⇒ "*PrISp*".

5.30. Tähistagu antud tekstis mingi osa paigutamine ümarsulgudesse, et see tekstiosa esineb järjest kaks korda.

Koostada uus tekst, mis saadakse antud tekstist sulgude avamise teel (sulgude sisu topeldades).

Näiteks,
"*m(a)hä(r)a (läki)*" ⇒ "*maahärra läkiläki*";
"*(a)bc(d(e))*" ⇒ "*aabcdeedee*".

Täpsemalt, sulgude avamise teisenduseks T (tekstis σ) on

$$T(\sigma) = \begin{cases} \sigma, \text{ kui tekst } \sigma \text{ ei sisalda suluavaldist,} \\ T(\alpha)T(\beta)T(\beta)T(\gamma), \text{ kui } \sigma = \alpha(\beta)\gamma. \end{cases}$$

5.31. Mittetühi tekst (sõnad ja tühikud) on antud ridade (sõnede) järjendina. Leida selle teksti realõpu sõnadest pikima pikkus.

Näiteks, kolme alljärgneva teksti korral:

1) "*Aias sadas saia*"
   "*Kuu paistab tähed säravad*"
2) "*Me õpime*"
   "*astronoomiat*"
   "*meeleldi*"
3) "*Ruumist väljas käia* "
   "*enne enda töö sooritamist* "
   "*lubatakse ainult eriloaga* "
   " "
   " *erandkorras* "

on vastusteks 1):7; 2):12; 3):11.

5.32. Antud teksti t (sõnad ja tühikud) ning arvu k korral koostada uus tekst u, mis koosneb tühikutega eraldatud teksti t šifreeritud sõnadest. Sõna šifreering saadakse sõna tähtede tsüklilisel ümberpaigutamisel $|k|$ koha võrra – ettepoole (paremale), kui $k > 0$ ja tahapoole (vasemale), kui $k < 0$.

Näiteks,

- kui $t =$" *Hello world* " ja $k = -3$, siis $u =$"*loHel ldwor*";
- kui $t =$" *vanakuradi vanaema*" ja $k = 6$, siis $u =$"*kuradivana anaemav*";
- kui $t =$" *ah* " ja $k = 3$, siis $u =$"*ha*".

5.33. Loetleda kõik vähemalt 5-tähelised sõnad, mis antud tekstis esinevad vähemalt 10 korda.

5.34. *Numbrisõne* koosneb numbritest ja ümarsulgudest. Numbrisõne väärtus on tema numbrite summa, kuid sulgudesse paigutamine muudab väärtuse vastasmärgiliseks.

Leida antud numbrisõne väärtus.

Näiteks,

numbrisõne "123(45)" väärtuseks on -3 (arvutuskäik $1+2+3-(4+5)$);

numbrisõne "(1)2(3(4)5)" väärtuseks on -3 (arvutuskäik $-1+2-(3-4+5)$).

5.35. Koostada sõne s', mille saab antud sõnest s nende suurte ja väikeste eesti tähtede asendamisel punktiga, mille väikevaste järjekorranumber eesti tähestikus (*abcdefghijklmnopqrsšzžtuvwõäöüxy*) on suurem antud piirist p.

Näiteks,

kui $s =$"*HALLOO kosmos!*" ja $p = 13$, siis $s' =$"*HALL.. k..m..!*";

kui $s =$"*vanakuradi666vanaema*" ja $p = 1$, siis $s' =$"*.a.a...a..666.a.a..a*".

5.36. Teisendada antud sõna s selliselt, et sõna s iga täishääliku t järele lisatakse tähepaar "*pi*" tingimusel, et

t ei ole sõna s viimane täht,

ja

t on sõna s esimene täht või talle eelneb kaashäälik sõnas s.

Näiteks,

"*mina*" $\Rightarrow$ "*mipina*",

"*paljud*" $\Rightarrow$ "*papiljupid*",

"*kodumaa*" $\Rightarrow$ "*kopidupimapia*",

"*Ants*" $\Rightarrow$ "*Apints*".

5.37. Antud teksti (sõnad ja tühikud) korral leida esimesed kaks järjestikku paiknevat võrdse pikkusega sõna ja nende erinevuste arv. Erinevuseks nimetame positsiooni, millel neis sõnades paiknevad erinevad tähed.

Näiteks,

tekstis "*Teatavasti kord Miki läks Piritalt merele*" on esimesed kaks järjestikust võrdse pikkusega sõna "*kord*" ja "*Miki*"; nendel on 4 erinevust (kõigil neljal positsioonil asuvad erinevad tähed);

tekstis "*läki lausa lahingusse Läki läki*" on esimesed kaks järjestikust võrdse pikkusega sõna "*Läki*" ja "*läki*"; nendel on 1 erinevus (esimesed tähed on erinevad);
tekstis "*hakkame mehed minema*" ei leidu järjestikuseid ühepikkuseid sõnu.

5.38. Antud tekstist eemaldada kõik suluavaldised (ümarsulgudega piiratud osad).

Näiteks,

"$abc(def)g(h)i$" $\Rightarrow$ "$abcgi$";

"$a(b((c)d))e((ab)(cd))$" $\Rightarrow$ "ae";

5.39. Koostada sõne, mis saadakse antud sõnest selle väikestest tähtedest 's' ja 'z' koosnevate osade kõigi tähtede asendamisel vastava osa esimese tähega.

Näiteks,
"*Meesz, kesz teadisz uzzzszsziszzsõnu*" → "*Meess, kess teadiss uzzzzzzzzissssõnu*";
"*FIDESz*" → "*FIDESz*".

5.40. Leida palindroom-sõnade arv antud tekstis (sõnad ja tühikud), suur- ja väiketähti eristamata.
Näiteks,
tekstis "*Aias sadas alla sajas sai*" on 3 palindroom-sõna;
tekstis "*Igikaasajaloosalasoolajasaakigi ei läinud vaja*" on neid 1.

5.41. Antud suluavaldisi sisaldavas tekstis eemaldada paarituarvulise sügavusega (sulgudesse sisestatuse mõttes) sümbolid ja kõik sulud.
Näiteks,
"$abc(def)g(h)i$" ⇒ "$abcgi$";
"$a(b((c)d))e((ab)(cd))$" ⇒ "$adeabcd$";
"$(abc(de(fg))h)$" ⇒ "de".

5.42. Antud sõnes asendada kõigis vaid suurtähtedest koosnevates (pikimates) alamsõnedes suurtähed vastavate väiketähtedega, alates alamsõne teisest tähest.
Näiteks,
"*Kanter lendas USAsse tRRRReenima.*" ⇒ "*Kanter lendas Usasse tRrrreenima.*"

5.43. Antud tekstis asendada iga suurtähega algav sõna sama pika sõnaga, mille esimeseks täheks on X ja järgmisteks x.
Näiteks,
"*EV, kohtuotsus Agu Sihvka süüasjas.*" ⇒ "*Xx, kohtuotsus Xxx Xxxxxx süüasjas.*"

5.44. Antud on eestikeelne tekst, milles tühikud ei paikne päris korrektselt. Korrastada see tekst, nii et oleksid täidetud nõuded:

1) kirjavahemärgi (.?!,;:) ees tühikut ei ole,
2) iga tekstisisese kirjavahemärgi järel on parajasti üks tühik,
3) kahe järjestikuse sõna vahel on parajasti üks tühik,
4) ning teksti alguses ega lõpus tühikuid ei ole.

Näiteks, teksti
" *Ta küsis , kas tohib . Kes küsis ?Tema küsis - järelikult mitte sina . "*
korrastamisel saadakse
"*Ta küsis, kas tohib. Kes küsis? Tema küsis - järelikult mitte sina .*"

5.45. Lisada antud sõnes iga kordusgrupi sees iga kahe sümboli järele sidekriips. (Mõiste *kordusgrupp* – vt ülesanne 5.13.)
Näiteks,
"*Aaaaarrghhhh*" ⇒ "*Aa-aa-arrghh-hh*".

5.46. Antud teksti kõigis väiketähti sisaldavates sõnades asendada suurtähted vastavate väiketähtedega.

Näiteks,

"*USA ja Vene ja riigiPead poksiringis USAs.*" ⇒ "*USA ja vene ja riigipead poksiringis usas.*"

5.47. Koostada sõne, milles on antud sõne kõigi sulgevate sulgude indeksid (tühikutega eraldatult) vastavate avavate sulgude esinemise järjekorras.

Näiteks, sõnele "*(a)bc(d(e))*" vastavalt koostatud sõne on "*2 10 9*".

5.48. Koostada sõne kahe antud sõne ühistest sümbolitest nende järjestust muutmata. Ühiseks loetakse siin sümbol, mis esineb mõlemas sõnes sama indeksiga kohal. Tõstutundetu.

Näiteks, sõnede "*abrakadabraaaa*" ja "*Babrak Karmal*" ühissümbolite sõneks on "*ra*".

5.49. Antud tekstis eemaldada numbrid kõigis tähti sisaldavates sõnades.

Näiteks, "*Pelle õel Pille15l oli 15 palli.*" ⇒ "*Pelle õel Pillel oli 15 palli.*"

5.50. Loetleda antud väiketähtedest koosneva sõna tähtede kõik sellised permutatsioonid, kus ükski kolm järjest paiknevat tähte ei ole tähestikuliselt järjestatud (ei mittekahanevalt ega ka mittekasvavalt).

Näiteks, sellised permutatsioonid sõne "*karu*" korral :

kaur, *krau*, *kuar*, *arku*, *aukr*, *rauk*, *rkua*, *ruak*, *uark*, *ukra*.

5.51. Loendada antud sõne vähemalt pikkusega 2 alamsõned, millel on sama algus- ja lõpusümbol.

Näiteks, sõnes "*baazba*" on selliseid alamsõnu 4: "*baazb*", "*aazba*", "*aa*", "*azba*".

5.52. Kontrollida, kas üks antud sõne on saadud teisest antud sõnest selle sümbolite tsüklilise nihutamise teel.

5.53. Antud on kaks ühepikkust heterogrammi s ja t.

Olgu Y nende ühissümbolite hulk (sümbolite hulk, mis esinevad nii sõnes s kui ka sõnes t).

Leida kaks arvu x ja y:

x = nende ühissümbolite ($\in Y$) arv, mis asuvad sõnes s ja sõnes t samal kohal;

y = nende ühissümbolite ($\in Y$) arv, mis asuvad sõnes s ja sõnes t erinevatel kohtadel.

Näiteks,

s ="*pesukorv*",

t ="*vasikäpe*";

$Y = \{$*'p', 'e', 's', 'k', 'v'*$\}$;

$x = 2$ (sest *'s'* ja *'k'* asuvad kohakuti);

$y = 3$.

5.54. Kontrollida, kas antud arvu esitus kaheksandsüsteemis on palindroom.
Näiteks,
arvu 495 esitus kaheksandsüsteemis 757_8 on palindroom;
arvu 454 esitus kaheksandsüsteemis 706_8 ei ole palindroom.

5.55. Antud sõnes leida pikim palindroom-alamsõne.

5.56. Antud sõnes leida pikim erinevatest sümbolitest koosnev alamsõne.

5.57. Antud on sõned s ja t. Loetleda sõne s need alamsõned, mis on sõne t anagrammid.

5.58. Rooma numbrite süsteemis kasutatakse arvude 1, 5, 10, 50, 100, 500 ja 1000 tähistamiseks vastavalt tähti I, V, X, L, C, D ja M. Täisarvud $1\ldots9$ kirjutatakse järgmiselt: I, II, III, IV, V, VI, VII, $VIII$ ja IX. Analoogiliselt tähistatakse kümnelised ja sajalised. Näiteks, 1941 = MCMXLI.

(a) Antud arv vahemikust (0; 3900) esitada Rooma numbrite süsteemis.

(b) Leida arv, mille tähis on antud Rooma numbrite süsteemis.

5.59. Leida antud tuhandest väiksema naturaalarvu esitus sõnalisel kujul eesti keeles.

5.60. Antud on täisarv n vahemikust $(-120;160)$. Loetleda kõik võimalused selle arvu esitamiseks avaldisena, mis on saadud, kui numbrite vahele reas 123456789 on lisatud kas plussmärk, miinusmärk või mitte midagi.
Näiteks, arvu $n = 151$ jaoks leidub 3 võimalust:
$1-2-3+4-5+67+89=151$,
$12-3+4+56-7+89=151$,
$123-4+56-7-8-9=151$,
arvu -100 jaoks aga ainult 1 võimalus:
$1-2+3-4-5-6-78-9=-100$.

5.61. Antud on 10 sõna. Koostada pikim sõnade järjend, kus järgmise sõna esitäht on sama, mis eelmise viimane täht.
Näiteks, sõnade "*abielu*", "*ema*", "*isa*", "*laps*", "*onu*", "*tädi*", "*lõpp*", "*uus*", "*tool*", "*aga*" korral on tulemuseks järjend ("*tädi*", "*isa*", "*aga*", "*abielu*", "*uus*").

5.62. Sümbolite "varu" on antud sõnena s. Koostada (üks) pikim palindroom, kasutades sümboleid sõnest s.
Näiteks, sõne "*acbcdbae*" sümbolitest saab koostada palindroomi "*abcecba*".

5.63. Leida (kui võimalik) esimene unikaalne sümbol antud sõnes.
Näiteks, sõnes "*agcdbachhg*" on esimeseks unikaalseks sümboliks '*d*'.

5.64. Antud sõnest eemaldada korduvalt kordusgrupid, kuni alles ei jää enam ühtegi kordusgruppi. (Mõiste *kordusgrupp* – vt ülesanne 5.13.)
Näiteks, "*dbaabdab*" $\Rightarrow$ "*ab*".

5.65. Antud sõnes jätta igast kordusgrupist alles ainult üks sümbol.
Näiteks, "*aabbbcdd*" $\Rightarrow$ "*abcd*".

5.66. Antud on mingi tekst ja arv k. Loetleda k selles tekstis kõige sagedamini esinevat sõna.

5.67. Transponeerida antud tekst (sõnad ja tühikud).
Näiteks, "*Ants läks kooli*" $\Rightarrow$ "*kooli läks Ants*".

5.68. Loetleda antud tekstis esinevad anagrammide rühmad.
Näiteks, tekstist "*Elvi isa laual on tore asi - telefon. No kaubajaam sai remoditud, kuid kahjuks mitte kaubamaja; asi pole korras. Laeva vile lõi majaka kajama.*" leiame järgmised anagrammide rühmad
("*elvi*", "*vile*")
("*on*", "*no*")
("*majaka*", "*kajama*")
("*asi*", "*sai*", "*isa*")
("*kaubajaam*", "*kaubamaja*").

5.69. Funktsioon sümbolite hulgast S sümbolite hulka S', $kod : S \to S'$ on *kodeerimisfunktsioon*, kui $c \neq c' \Rightarrow kod(c) \neq kod(c')$. Kaks sõnet $s = s_0 s_1 \ldots s_{m-1}$ ja $t = t_0 t_1 \ldots t_{n-1}$ on *isomorfsed*, kui $m = n$ ja leidub kodeerimisfunktsioon kod nii, et kui $t_i \neq s_i$, siis $kod(s_i) = t_i$ $(i = 0, 1, \ldots n-1)$.
Kontrollida, kas kaks antud sõnet on isomorfsed.
Näiteid:

- sõned $s =$"*abc*" ja $t =$"*dek*" on isomorfsed;
  kodeerimisfunktsiooniks $kod : \{a,b,c\} \to \{d,e,k\}$ sobib
  $a \to d, b \to e, c \to k$
- sõned $s =$"*eggg*" ja $t =$"*addg*" on isomorfsed;
  kodeerimisfunktsiooniks $kod : \{e,g\} \to \{a,d\}$ sobib
  $e \to a, g \to d$
- sõned $s =$"*title*" ja $t =$"*paper*" on isomorfsed;
  kodeerimisfunktsiooniks $kod : \{e,i,l,t\} \to \{r,a,e,p\}$ sobib
  $e \to r, i \to a, l \to e, t \to p$
- sõned $s =$"*titlee*" ja $t =$"*papera*" ei ole isomorfsed.

5.70. Arvutada

(a) Poola pöördkujul ehk postfikskujul

(b) Poola kujul ehk prefikskujul

sõnena antud aritmeetilise avaldise väärtus.

5.71. Kontrollida, kas antud suluavaldis

(a) on sulgude paigutuselt korrektne;

(b) ei sisalda tarbetuid sulukordusi.

Näiteks, suluavaldis $(A\alpha(\underline{(}(B\gamma[2\beta C])\underline{)}))$ on sulgude paigutuselt korrektne, kuid sisaldab tarbetut ümarsulukordust.

5.72. Koostada antud sõne kõikide sufiksite kahaneva pikkusega loetelu leksikograafiliselt sorteeriv permutatsioon.

Näiteks, sõne "*kanala*" jaoks on see $(5,3,1,0,4,2)$, sest selle sõna sufiksite loetelu pikkuse kahanedes on ("*kanala*", "*anala*", "*nala*", "*ala*", "*la*", "*a*"). Viimase ümberkorraldus vastavalt leitud permutatsioonile on ("*a*", "*ala*", "*anala*", "*kanala*", "*la* ", "*nala*").

(Sõne *sufiksiteks* on selle lõpus paiknevad alamsõned.)

5.73. Loetleda antud sõnaga s kõige paremini riimuvad sõnad antud sõnade hulgast S, arvestades et

(a) kaks sõna on seda riimuvamad, mida rohkem järjestikuseid tähti lõpust lugedes neil kokku langeb; seejuures võib ühe sõna (päris)sufiks võrduda terve teise sõnaga;

(b) riimuvad sõnad on need, milles esinevad täpselt samad täishäälikud täpselt samas järjekorras, kusjuures täishäälikute vahel on vähemalt üks kaashäälik.

Näiteks (juht a),
sõnaga $s =$"*programm*" riimuvad parimini hulga $S =$ {"*asulavesi*", "*karuramm*", "*kiirtramm*", "*pesulavesi*", "*plastmasskann*", "*programm*","*tantsusamm*","*tiritamm*", "*tramm*"}
sõnad
"*karuramm*", "*kiirtramm*" ja "*tramm*".

5.74. Antud sõna s korral arvutada selle järjekorranumber kõigi leksikograafiliselt sorteeritud s permutatsioonide hulgas,

(a) kui sõna s on heterogramm;

(b) kui sõnas s on korduvaid sümboleid.

Näiteks (juht a), $s =$ "*bac*" järjekorranumber hulgas {"*abc*", "*acb*", "*bac*", "*bca*", "*cab*", "*cba*"} on 3.

5.75. Mängu "paber-kivi-käärid" kulg on antud ühepikkuste sõnejärjenditena a ja b, mille elemendid on hulgast {"*paber*", "*kivi*", "*käärid*"}. Ühe ja sama indeksiga (i) sõnede paar (a_i, b_i) vastab ühele (i-ndale) voorule, mil esimene mängija näitas sõna a_i ja teine – sõna b_i.

Leida selle mängukäigu skoor $m : n$, kus m on esimese mängija võidetud voorude arv ja n – teise mängija võidetud voorude arv.

Näiteks,
$a =$ ("*käärid*", "*paber*", "*kivi*", "*käärid*", "*paber*", "*kivi*", "*kivi*")

$b =$ (“*kivi*”, “*kivi*”, “*paber*”, “*käärid*”, “*kivi*”, “*käärid*”, “*kivi*”)
korral on skoor 3 : 2 esimese mängija kasuks.

5.76. *Kontrollsõne* on heterogramm, mis määrab temasse kuuluvate sümbolite lubatud esinemisjärjestuse mõnes teises vaadeldavas sõnes.

Olgu antud kontrollsõne s_0. Siis kahesümboliline sõne “c_1c_2” on *lubamatu* selle kontrollsõne mõttes, kui sõnes s_0 leidub osasõne “c_2c_1”.

Kontrollida, kas antud sõne s sümbolid on kontrollsõnega s_0 lubatud järjestuses, st kas selles ei leidu lubamatuid (kahesümbolilisi) osasõnesid.

Näiteks,
kui $s_0 =$“*aeiouõäöü*”, siis
sõne $s =$“*kontrollsõne*” sümbolid ei ole kontrollsõnega s_0 lubatud järjestuses,
sõne $s =$“*aedpeedipõld*” sümbolid aga on kontrollsõnega s_0 lubatud järjestuses.

II Rekursiivselt

5.77. Transponeerida antud sõne.

5.78. Loetleda antud sõne kõik osasõned.

5.79. Leida antud sõnade pikim ühisprefiks.
(Sõne *prefiksiteks* on selle alguses paiknevad alamsõned.)
Näiteks, sõnade “*kaevutee*”, “*kaevur*”,“*kaevik*” ja “*kaevama*” pikim ühisprefiks on “*kaev*”.

5.80. Antud on n sümbolist koosnev sõne ja arv k $(0 < k < n)$. Loetleda kõik sõned, mis saadud k sümboli kustutamisel antud sõnest (st kõik osasõned pikkusega $n-k$).

5.81. Loetleda antud sõne kõik variatsioonid antud arvu k kaupa (kõik k-permutatsioonid).
Näiteks, sõne “*karu*” ja $k = 3$ korral
“*kar*”, “*kra*”, “*akr*”, “*ark*”, “*rak*”, “*rka*”,
“*kau*”, “*kua*”, “*aku,*” “*auk,*” “*uka*”, “*uak*”,
“*kru*”, “*kur*”, “*rku*”, “*ruk*”, “*ukr*”, “*urk*”,
“*aru*”, “*aur*”, “*rau*”, “*rua*”, “*uar*”, “*ura*”.

5.82. Loendada sõnas esinevad konsonandid.

5.83. Loendada antud sõne sümbolite

(a) kõikvõimalikud rühmitused;

(b) kõik sellised rühmitused, milles iga rühm on palindroom.

Rühmadeks on antud sõne mittelõikuvad järjestikused alamsõned.
Näiteks,
sõne “*abc*” sümbolite rühmitusi on kokku 3 (juht a):

“*a*” | “*b*” | “*c*”,
“*a*” | “*bc*”,
“*ab*” | “*c*”,
“*abc*”;
sõne “*abaa*” selliseid rühmitusi, milles iga rühm on palindroom, on kokku samuti 3 (juht b):
“*a*” | “*b*” | “*a*” | “*a*”,
“*a*” | “*b*” | “*aa*”,
“*aba*” | “*a*”.

5.84. Loetleda kõik antud sõnedejärjendi a jaotused kahte mittetühja rühma. Iga jaotus esitatakse järjendi a kahe mittelõikuva osajärjendina.

Näiteks,
järjendi $a =$(“*ab*”, “*cde* ”, “*fg*”) kaheks-jaotused on
(“*ab*”) (“*cde*”, “*fg*”),
(“*cde*”) (“*ab*”, “*fg*”),
(“*fg*”) (“*ab*”, “*cde*”);
järjendi $a =$(“*ab*”, “*cde* ”, “*fg*”, “*h*”) kaheks-jaotused on
(“*ab*”) (“*cde*”, “*fg*”, “*h*”),
(“*cde*”) (“*ab*”, “*fg*”, “*h*”),
(“*fg*”) (“*ab*”, “*cde*”, “*h*”),
(“*h*”) (“*ab*”, “*cde*”, “*fg*”),
(“*ab*”, “*cde*”) (“*fg*”, “*h*”),
(“*ab*”, “*fg*”) (“*cde*”, “*h*”),
(“*ab*”, “*h*”) (“*cde*”, “*fg*”)

5.85. Antud sõne koosneb sümbolitest `'0'`,`'1'` ja `'?'`. Loetleda kõik bitisõned, mis on saadud sellest sõnest iga sümboli `'?'` asendamisel kas sümboliga `'0'` või sümboliga `'1'`.

Näiteks, küsimärkide asendamisega sõnes
“??11?00” saadakse bitisõned:
“0011000”
“0011100”
“0111000”
“0111100”
“1011000”
“1011100”
“1111000”
“1111100”.

5.86. Loetleda antud sõne kõik permutatsioonid

(a) suvalises järjestuses;

(b) leksikograafilises järjestuses.

5.87. Antud sõne s korral

(a) leida vähim sümbolite arv, mille eemaldamisel sõnest s saadakse palindroom;

(b) loetleda pikimad palindroomid, mis on saadud mõnede sümbolite eemaldamisel sõnest s.

Näiteks, sõnest "*acbcdbae*" saadakse pikimad palindroomid "*abdba*", "*abcba*" ja "*acbca*", kolme sümboli eemaldamisel vastavalt kohtadest $(2,4,8)$, $(2,5,8)$ ja $(5,6,8)$.

5.88. Antud on n sõnajärjendit. Loetleda (pikkuse järjestuses) kõik laused, mis saadakse, kui võtta üks sõna igast järjendist.

Näiteks, kolme sõnajärjendi
("*Mari*", "*Jüri*"), ("*jookseb*", "*joob*") ja ("*kiiresti.*", "*terviseks.*", "*mõnuga.*")
puhul saadakse laused
"*Mari jookseb terviseks.*"
"*Jüri jookseb terviseks.*"
"*Mari jookseb kiiresti.*"
"*Jüri jookseb kiiresti.*"
"*Mari jookseb mõnuga.*"
"*Mari joob terviseks.*"
"*Jüri jookseb mõnuga.*"
"*Jüri joob terviseks.*"
"*Mari joob kiiresti.*"
"*Jüri joob kiiresti.*"
"*Mari joob mõnuga.*"

5.89. Koostada antud sõne permutatsioonide hulk.

Näiteks,
sõne "*abc*" permutatsioonide hulk on {"*abc*", "*acb*", "*bac*", "*bca*", "*cab*", "*cba*"};
sõne "*aab*" permutatsioonide hulk on {"*aab*", "*aba*", "*baa*"}.

5.90. Antud on kaks mittetühja sõnade järjendit.

Loetleda kõik nende järjendite *põimingud* – laused, mis sisaldavad mõlema sõnajärjendi kõik elemendid nii, et kummagi järjendi sõnade omavaheline järjestus pole muutunud. (Vt ka ülesanne 4.67.)

Näiteks, kui on antud järjendid, ("*kas*", "*mina*") ja ("*olen*", "*siin*"), siis saadakse lausete loeteluks
"*kas mina olen siin*",
"*kas olen mina siin*",
"*kas olen siin mina*",
"*olen kas mina siin*",
"*olen kas siin mina*",
"*olen siin kas mina*".

5.91. Kontrollida, kas antud sõne on kahe teise antud sõne sümbolite põiming.

Näiteks, sõne "*dabec*" on sõnede "*abc*" ja "*de*" põiming.

5.92. Sõne iga osasõne on määratud ühe indeksikomplektiga – sõne sümbolite indeksitega, millistelt kohtadelt sõnest on sümbolid sellesse osasõnesse valitud.

Näiteks, sõnest $s =$"*dabecac*" saame osasõned "*dac*", "*ba*" ja "*daba*", valides sümbolid sõnest s vastavalt indeksitelt $\{0,1,4\}$, $\{2,5\}$ ja $\{0,1,2,5\}$.

Pikkusega n sõne kõikvõimalike osasõnede indeksikomplektide hulk on indeksite hulga $\{0,1,\ldots n-1\}$ alamhulkade hulk (ilma tühja alamhulgata).

Antud sõne s korral leida (kui võimalik) kaks pikimat mittelõikuvat indeksikomplekti, millega valitakse kokkulangevad osasõned (st kaks võrdset osasõne saadakse sõne s erinevatest kohtadest valides).

Näiteks,

sõne $s =$"*adabcb*" korral on vastuseks indeksikomplektid $\{0,3\}$ ja $\{2,5\}$, millele mõlemale vastab osasõne "*ab*";

sõne $s =$"*abacbdefd*" korral on vastuseks indeksikomplektid $\{0,1,5\}$ ja $\{2,4,8\}$, millele mõlemale vastab osasõne "*abd*".

5.93. Antud sõne koosneb numbritest ja ümarsulgudest.

Leida selle sõne väärtus kui numbrite summa, arvestades, et

(a) sulgudesse paigutamine tähendab ruututõstmist;

(b) sulgudesse paigutamine muudab väärtuse märgi vastupidiseks.

Näiteks,

juht a:

sõne "*(12)345*" väärtuseks on 21 $(= (1+2)^2+3+4+5)$;

sõne "*1(2)(3(45))*" väärtuseks on 7061 $(= 1+2^2+(3+(4+5)^2)^2)$;

juht b:

sõne "*(1)2(3(4)5)*" väärtuseks on -3 $(= -1+2-(3-4+5))$.

5.94. Aritmeetiline avaldis on antud sõnena, milles

- tehtemärgid on hulgast { '+', '-', '*', '/' };
- kõik operandid on nullist erinevad kümnendnumbrid;
- iga tehte operandide ümber on täpselt ühed ümarsulud;
- kogu avaldis on ümarsulgudes.

Leida antud aritmeetiline avaldise väärtus.

Näiteks, $(((3)-(4))/((5)+(3))) = -0.125$.

5.95. Ühes programmeerimiskeeles loetakse kommentaariks ümarsulgudesse paigutatud tekst. Kommentaari sees olev ümarsulgudes tekst aga loetakse koodiks, milles omakorda võib olla kommentaare jne.

Antud sõnest eemaldada kommentaarid.

Näiteks,

"*abc(def)g(h)i*" $\Rightarrow$ "*abcgi*";

"*a(b((c)d))e((ab)(cd))*" $\Rightarrow$ "*adeabcd*";

"*(abc(de(fg))h)*" $\Rightarrow$ "*de*".

5.96. Antud on lähtesõne s pikkusega $|s| = n$ ja naturaalarv k. Sõne s sümbolitest on kogu aeg "nähtaval" k esimest sümbolit (või vähem, kui k on suurem, kui lähtesõne jooksev pikkus). Moodustada sama pikk tulemussõne s' , mis on lähtesõnest saadud järgmise operatsiooni korduval rakendamisel:

- Leida sõne s "nähtavate" sümbolite seast vähim (vähima koodiga) sümbol ja suurim (suurima koodiga) sümbol ning paigutada need sõnest s ümber tulemussõne lõppu (vähim enne). Kui $|s| = 1$, siis paigutatakse ümber sõne s ainuke sümbol. Kui $|s| = 0$, siis lõpetatakse.

Näiteks,

lähtesõnest $s =$ *"kooskõlas"* saadakse $k = 3$ korral tulemussõne $s' =$ *"kokslõaso"* järgmiste sammudega:

1. $s =$"*kooskõlas*", $s' =$"";
2. $s =$"*oskõlas*", $s' =$"*ko*";
3. $s =$"*oõlas*", $s' =$"*koks*";
4. $s =$"*oas*", $s' =$"*kokslõ*";
5. $s =$"*o*", $s' =$"*kokslõas*";
6. $s =$"" $s' =$"*kokslõaso*".

5.97. Piirdume neljatäheliste eestikeelsete nimetavas käändes nimisõnade mingi alamhulgaga.

Antud lähtesõna ls ja sihtsõna ss korral leida (kui saab) võimalikult lühike sõnade järjend (tuletuskäik) $ls \to s1 \to s2 \to \ldots \to ss$, milles naabersõnad erinevad teineteisest parajasti ühe tähe poolest.

Näiteks,

sõnade $ls =$ *"kass"*, $ss =$ *"kakk"* üks tuletuskäike on

"kass" $\to$ *"kast"* $\to$ *"kart"* $\to$ *"kark"* $\to$ *"kakk"*,

kuid lühim on kahesammuline

"kass" $\to$ *"kaks"* $\to$ *"kakk"*.

5.98. Piirdume eestikeelsete nimetavas käändes nimisõnade mingi alamhulgaga S.

Antud lähtesõna ls ja sihtsõna ss korral leida (kui saab) võimalikult pikk sõnade järjend (tuletuskäik) $ls \to s1 \to s2 \to \ldots \to ss$, milles järgmine sõna $s' \in S$ on saadud eelmisest sõnast $s \in S$

kas

sõna s viimase tähe eemaldamisega,

või

sõnale s ühe tähe lisamisega,

või

sõnas s ühe tähe muutmisega.

Näiteks,

$S =$ {*"hiir"*, *"kass"*, *"pais"*, *"paisk"*, *"pass"*, *"piisk"*, *"kiir"*, *"viir"*, *"viis"*, *"viisk"*, *"raisk"*, *"kask"*} ja lähtesõna $ls =$*"hiir"* ning sihtsõna $ss =$*"kask"*

korral on pikimaks tuletuskäiguks
“hiir”→*“kiir”*→*“viir”*→*“viis”*→*“viisk”*→*“piisk”*→*“paisk”*→ *“pais”*→*“pass”*→*“kass”*→ *“kask”*.

6. Kahendpuude töötlemine

Kahendpuu tipus olevat kirjet nimetame *tipu märgendiks*. Arvulist märgendit nimetame ka *tipu võtmeks*. Erijuhul on märgendiks tühikirje.

Kasutame järgmisi lühendatud väljendeid:
"tipp M" tähenduses "tipp märgendiga M",
"tipus on M" tähenduses "tipu märgendiks/võtmeks on M",
"haru" tähenduses "alampuu".

Kahendpuu *tipu sügavuseks* on tee pikkus (tippude arv teel) juurtipust selle tipuni (juurtipu sügavuseks on 0). *Tasemeks* kahendpuus on ühe ja sama sügavusega tippude järjend (järjestuses vasakult paremale kahendpuus).

Kahendpuu kõrguseks on tee pikkus juurtipust viimase taseme tipuni.

Kahe alluvaga tippu nimetatakse *täistipuks*. Kahendpuu on *täis*, kui selles ei leidu ühe alluvaga tippe (st kõik vahetipud on täistipud).

Käesoleva jaotise joonistel on kahendpuud kujutatud töötlusprogrammi väljundi ekraanitõmmistena.

6.1. Luua

(a) juhuslik m tasemega kahendpuu;

(b) juhuslik m tasemega täis kahendpuu;

(c) täielik m tasemega kahendpuu (täis kahendpuu, mille kõik lehed asuvad viimasel tasemel);

(d) kompaktne n-tipuline kahendpuu (saadud täielikust kahendpuust null või enama lehe eemaldamisel viimase taseme lõpust);

(e) juhuslik n-tipuline kahendpuu;

(f) antud kahendpuu koopia (sellega sama struktuuriga ja samade tipumärgenditega kahendpuu);

(g) antud kahendpuu peegelpilt (vertikaaltelje suhtes);

(h) juhuslik m tasemega sümmeetriline kahendpuu (mis on iseenda peegelpilt).

6.2. Leida antud kahendpuu

(a) tippude arv;

(b) lehtede arv;

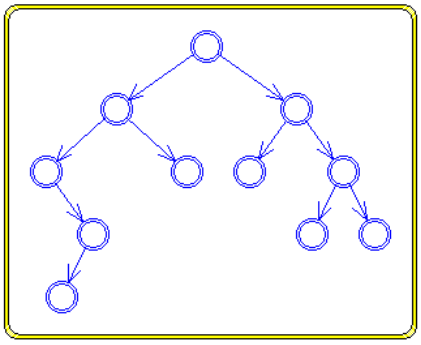
(c) vahetippude arv;

(d) täistippude arv;

(e) kõrgus;

(f) juure alampuude kõrguste aritmeetiline keskmine.

Vt näide joonisel 1.



```
Tippe: 11
Lehti: 5
Vahetippe: 5
Täistippe: 4
Kõrgus (tasemete arv): 5
Juure alampuude kõrguste keskmine: 3.5
```

Joonis 1: Kahendpuu ja selle arvulisi parameetreid.

6.3. Leida antud kahendpuu tihedus, so suhe $tippude\_arv/kõrgus$.

6.4. Kontrollida, kas antud kahendpuu

(a) on kompaktne;

(b) on täielik;

(c) on täis;

(d) on unikaalsete tipumärgenditega (st ei ole korduvaid märgendeid);

(e) on teise antud kahendpuuga sama struktuuriga;

(f) on teise antud kahendpuu koopia;

(g) on teise antud kahendpuu alampuu;

(h) on teise antud kahendpuu peegelpilt;

(i) on sümmeetriline.

6.5. Loetleda antud kahendpuu tipud

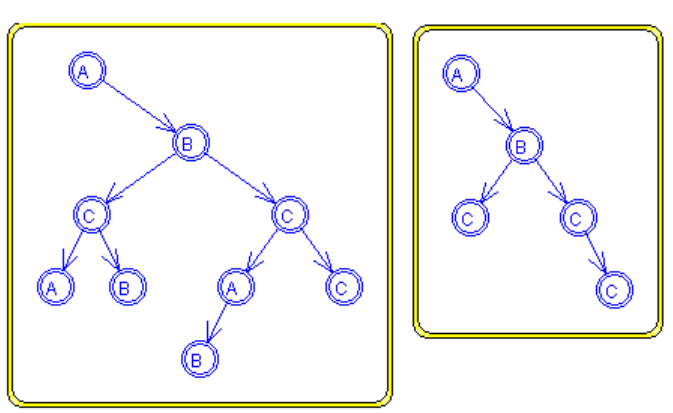
(a) eesjärjestuses;

(b) keskjärjestuses;

(c) lõppjärjestuses;

(d) tasemejärjestuses (st mööda tasemeid – vasakult paremale, ülalt alla).

6.6. Loetleda antud kahendpuu tasemed (tipujärjenditena).



Joonis 2: Kahendpuust (vasakul) lehttippude A ja B eemaldamistega saadud kahendpuu (paremal)

6.7. Antud kahendpuust eemaldada etteantud märgenditega lehti, kuni selliste märgenditega lehti enam ei ole. Vt näide joonisel 2.

6.8. Luua võimalikult madal kahendpuu, milles tipumärgenditeks on sümbolid Teie ees- ja perekonnanimest ning mille tippude lõppjärjestus annab Teie ees- ja perekonnanime.

6.9. Antud kahendpuusse lisada uus lehttipp võimalikult kõrgele – vähima sügavusega vabale kohale.

6.10. Antud kahendpuus leida antud tipule järgnev tipp

(a) eesjärjestuses;

(b) keskjärjestuses;

(c) lõppjärjestuses.

Vt näide joonisel 3.

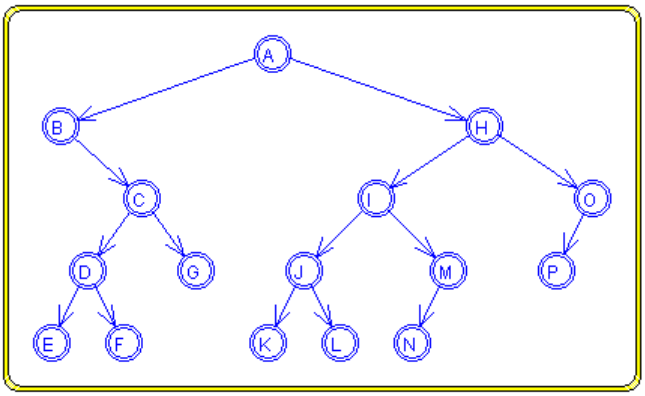
6.11. Loetleda antud kahendpuu tipud,

(a) mille alampuude kõrgused erinevad rohkem kui 1 võrra;

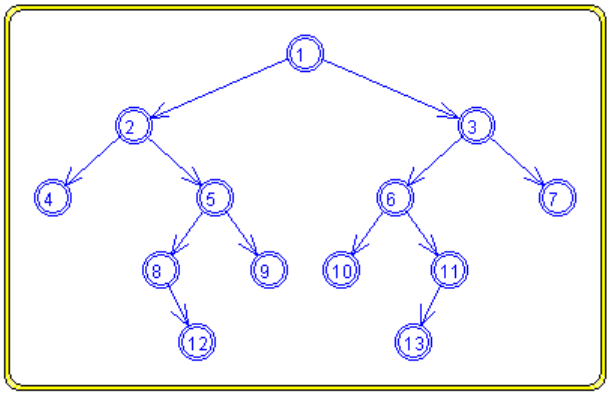
(b) mille alampuudes on võrdne arv tippe.

6.12. Kontrollida, kas antud kahendpuu on sümmeetrilise struktuuriga (sellise kahendpuu näide leidub joonisel 4).

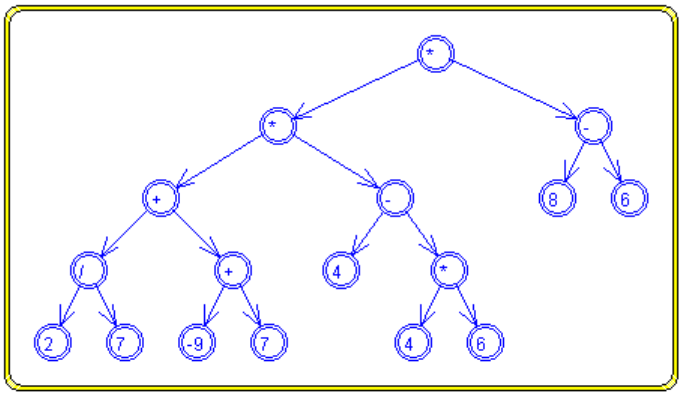
6.13. Luua m tasemega juhuslik *aritmeetilise avaldise puu* – täis kahendpuu, milles lehttippudes paiknevad arvulised operandid, juurtipul ja igal vahetipul on märgendiks binaarse tehte märk $+$, $-$, $*$ või $/$.



Joonis 3: Tipu G järglased ees-, kesk- ja lõppjärjestuses on vastavalt tipud H, A, C.



Joonis 4: Sümmeetrilise struktuuriga kahendpuu.



Joonis 5: Aritmeetilise avaldise kahendpuu ja selle väärtus ning vastav tavakujul avaldis.

6.14. Antud aritmeetilise avaldise puu korral

(a) leida vastava aritmeetilise avaldise väärtus;

(b) koostada antud aritmeetilise avaldise puule vastav tavakujul (infiks-kujul) aritmeetiline avaldis;

Vt näide joonisel 5.

6.15. Antud aritmeetilise avaldise puu koopias kirjutada iga vahetipu märgendiks sellest tipust algavale alampuule vastava aritmeetilise avaldise väärtus.
Vt näide joonisel 6.

6.16. Antud kahendpuu igasse tippu kirjutada selle tipu alluvate arv.

6.17. Antud kahendpuu iga tipu võtmele liita selle tipu vasaku haru tippude võtmete summa. Summeeritakse antud kahendpuus olnud võtmeid, mitte juba muudetuid. Vt näide joonisel 7.

6.18. Leida antud kahendpuu

(a) parempoolseim lehttipp;

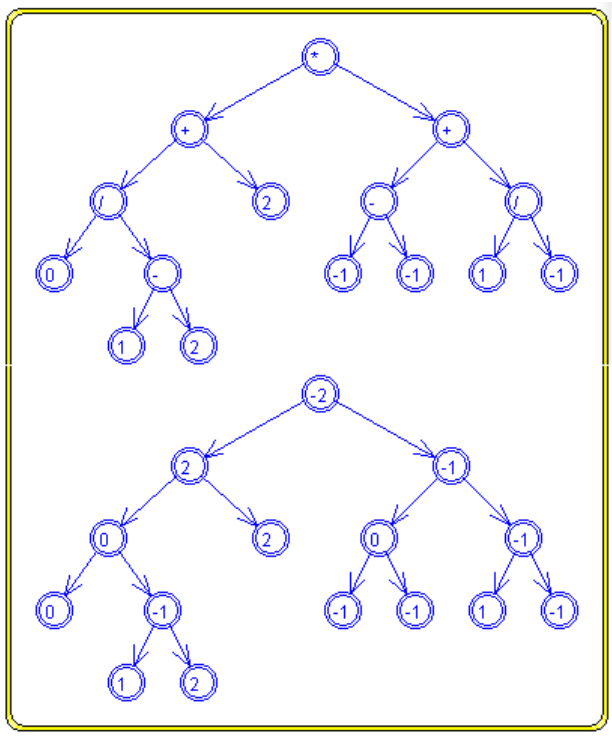
(b) vasakpoolseim lehttipp.

Vt näide joonisel 8.

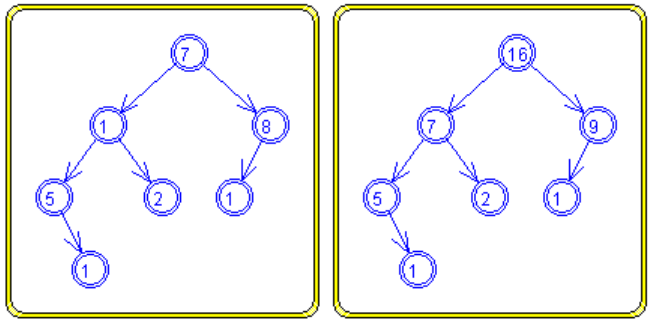
6.19. Loetleda antud kahendpuu

(a) parempoolse vaate tipud;

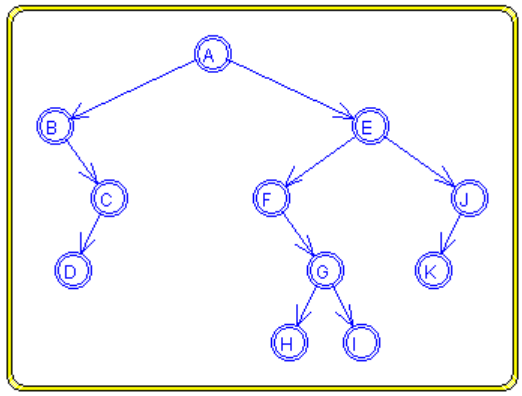
(b) vasakpoolse vaate tipud.



Joonis 6: Alumine saadud ülemisest ülesande 6.15 kohaselt.



Joonis 7: Tipuvõtmeid muudetud ülesande 6.17 kohaselt.



Joonis 8: Parempoolseimaks lehttipuks on K, vasakpoolseimaks – D.

Vasaku/parema vaate moodustavad tipud, mis on “nähtavad” vasakul/paremal asuvale vaatlejale. Vt näide joonisel 9.

6.20. Antud on kahendpuu ja selle kaks juurtipust erinevat tippu.

(a) Kontrollida, kas need tipud asuvad samal tasemel.

(b) Kontrollida, kas neil tippudel on sama vahetu ülemus.

(c) Leida nende tippude lähim ühine ülemus. Erijuhul, kui üks neist tippudest on teise (vahetuks või kaugemaks) ülemuseks, siis vastuseks lugeda kõrgemal asuv tipp.

6.21. Arvuliste (võtmetega) tippudega kahendpuu korral

(a) leida selliste tippude arv, mille võti on väiksem ülemuse võtmest;

(b) kontrollida, kas leidub kahe alluvaga tipp, mille võti võrdub alluvate võtmete aritmeetilise keskmisega;

(c) kontrollida, kas see on kahendotsimise puu (st iga tipu t vasakus harus ei ole suuremaid ja paremas harus väiksemaid võtmeid, kui on tipu t võti);

(d) leida (kui võimalik) kolm suurimat (erinevat) võtit;

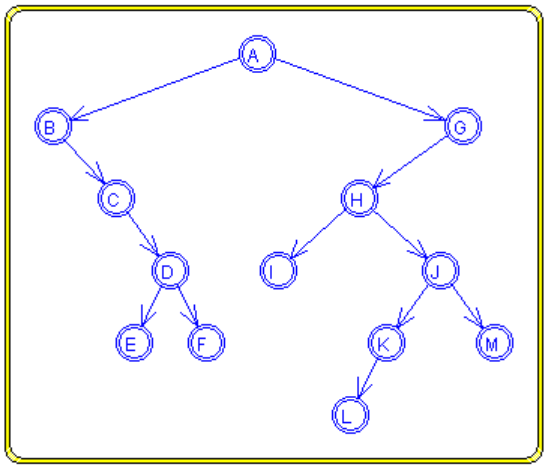
(e) leida suurim võtmete summa teekondadel juurtipust lehttippu;

(f) leida suurim tipu ja tema vahetu ülemuse võtmete erinevus;

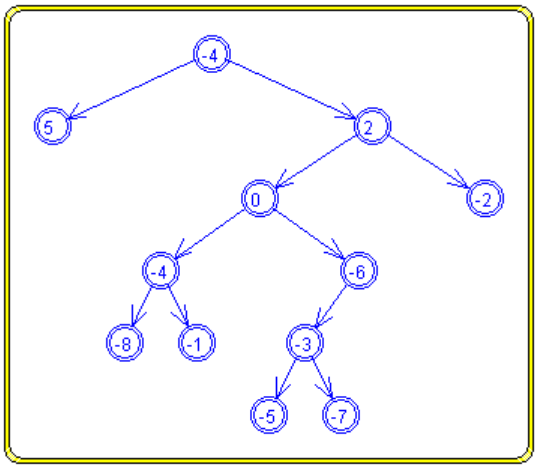
(g) muuta juurtipu ja vahetippude võtmed: tipu võtmeks omistada selle vahetute alluvate tippude võtmete summa;

(h) leida kõrgeima sellise alampuu juur, milles kõikide tippude võtmed on negatiivsed (vt näide joonisel 10).

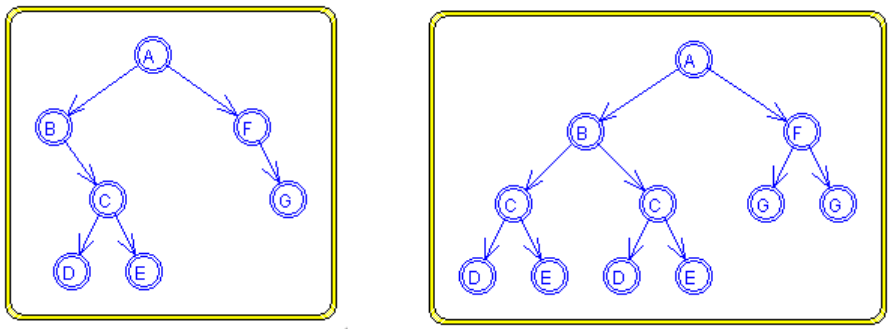
6.22. Loetleda antud kahendpuu tipud pikimal teel juurtipust lehttippu.



Joonis 9: Joonisel kujutatud kahendpuu vasak vaade on (A,B,C,D,E,L) ning parem – (A,G,H,J,M,L).



Joonis 10: Selles kahendpuus on tipust -6 algav alampuu suurima kõrgusega (3), ainult negatiivseid võtmeid sisaldav alampuu.



Joonis 11: Kahendpuu (vasakul) teisendamine täis kahendpuuks (paremal) ülesande 6.23 kohaselt.

6.23. Muuta antud kahendpuu täis kahendpuuks, lisades igale ühe alluvaga tipule teiseks alluvaks olemasoleva alluva täis kahendpuuks muudetud koopia.
Vt näide joonisel 11.

6.24. Loetleda (tippude järjenditena) kõik teed antud kahendpuus juurtipust lehttippu.

6.25. Antud on kahendpuu ja selle üks tipp.

(a) Loetleda tipud teel juurtipust selle tipuni.

(b) Loetleda tippude märgendid, mis korduvad teel juurtipust selle tipuni.

6.26. Antud kahendpuus lisada iga tipu märgendile selle tipu sügavus (sulgudes).

6.27. Leida antud kahendpuus

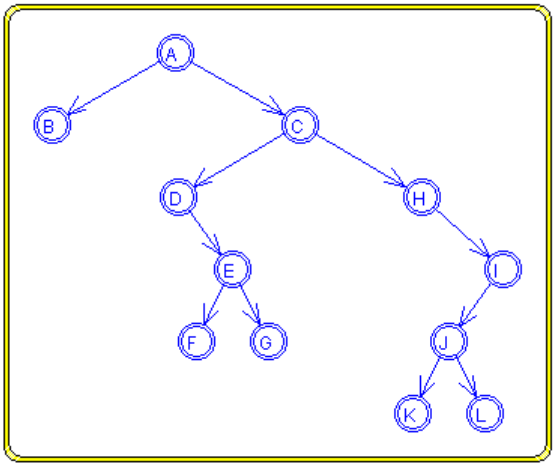
(a) kõrgeima lehttipu sügavus;

(b) antud tipule (üks) lähim alam-lehttipp.

Vt näide joonisel 12.

6.28. Luua uus kahendpuu, mis saadakse antud kahendpuust selle kõigi täistippude madalama haru äralõikamisel.
Vt näide joonisel 13

6.29. Antud kahendpuu kõigis tippudes paigutada suurema tippude arvuga haru vasakuks ja väiksema tippude arvuga haru paremaks alampuuks (võrdse tippude arvuga alampuid ei vahetata).
Vt näide joonisel 14.



Joonis 12: Kahendpuu kõrgeima lehttipu (B) sügavus on 1, tipu C lähim alamlehttipp on G.

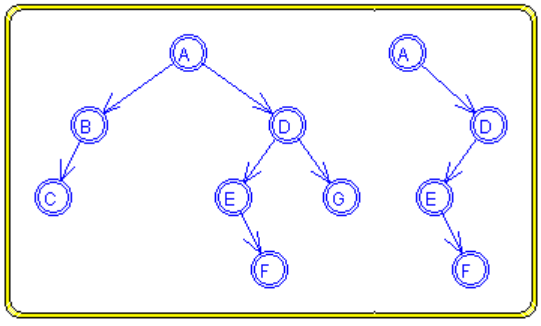
6.30. Kahendpuu *vasak raag* on tee puu juurtipust vasakpoolseima (st keskjärjestuses esimese) tipuni, *parem raag* – tee puu juurtipust parempoolseima (st keskjärjestuses viimase) tipuni.

Leida antud kahendpuu raagudest pikema tippudes olevate arvude korrutis. Kui raod on võrdse pikkusega, siis kasutada paremat raagu.

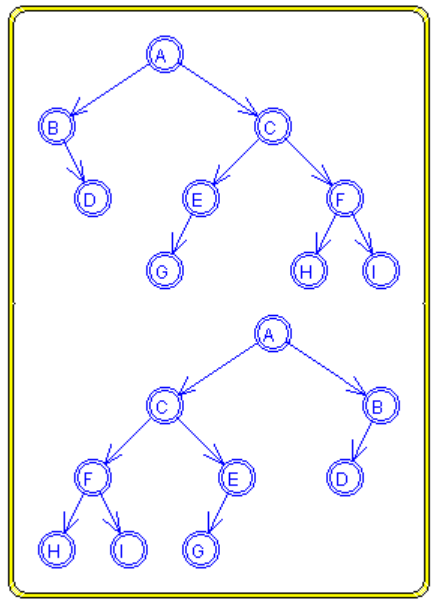
Vt näide joonisel 15.

6.31. Antud kahendpuu raagudele mittekuuluvate (nö “ülaltvaates kaetud”) tippude märgenditele lisada tärn.

Vt näide joonisel 16.



Joonis 13: Parempoolne saadud vasakpoolsest selle täistippude madalamate harude lõikamise teel.



Joonis 14: Alumine kahendpuu saadud ülemisest ülesande 6.29 kohaselt.

6.32. Luua uus kahendpuu, mis saadakse antud kahendpuust täpselt ühe alluvaga vahetippude ahelate “kokku tõmbamisel”, st ära kaotamisel, kusjuures võtmed kaotatava ahela tippudest liidetakse seda ahelat lõpetava tipu (täis- või lehttipu) võtmele.

Vt näide joonisel 17.

6.33. Leida antud kahendpuu vasaku haru parempoolseim tipp (kui vasak haru ei ole tühi) ja parema haru vasakpoolseim tipp (kui parem haru ei ole tühi) ning nende tippude sügavused antud kahendpuus.

Vt näide joonisel 18.

6.34. Leida antud kahendpuu pikimal juurtippu lehttipuga ühendaval teel asuvate tippude võtmete summa. Kui on mitu pikimat teed juurtipust lehttippudesse, siis valida see, mille lehttipp on vasakpoolseim.

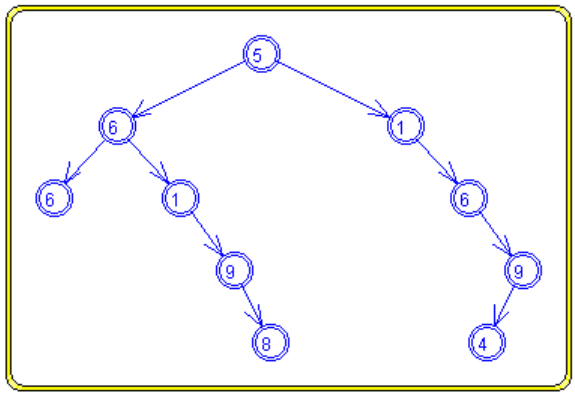
Vt näide joonisel 19.

6.35. Antud kahendpuus leida täpselt ühe vahetu alluvaga tippude arv.

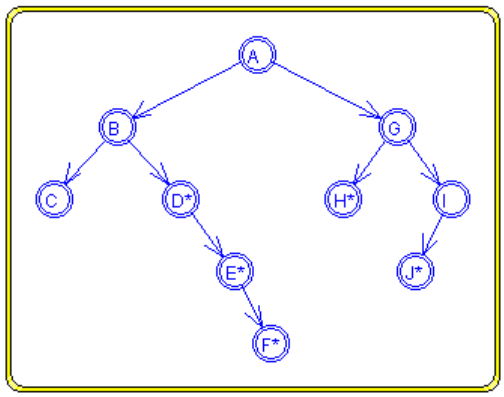
Vt näide joonisel 20.

6.36. Luua antud kahendpuu koopia, omistades iga tipu võtmeks suurema (võrdse) selle kahe alampuu tippude võtmetest.

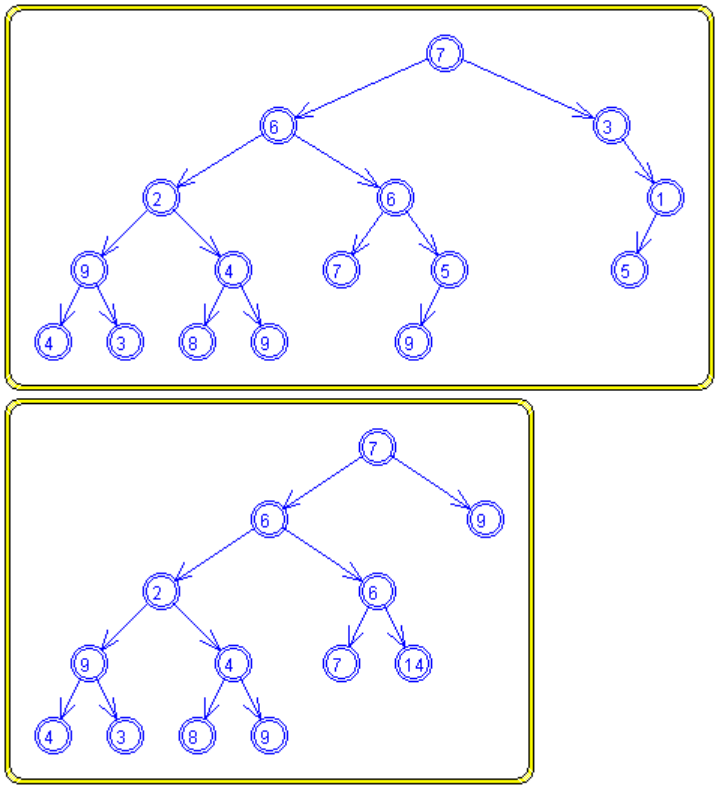
Vt näide joonisel 21.



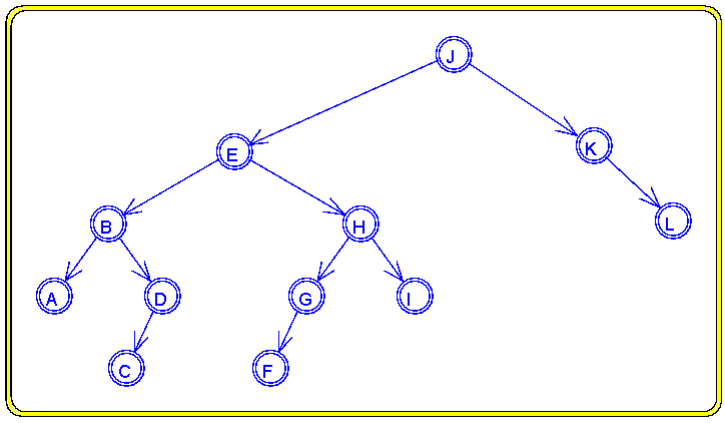
Joonis 15: Pikemal (parempoolsel) raol olevate arvude korrutis on 270.



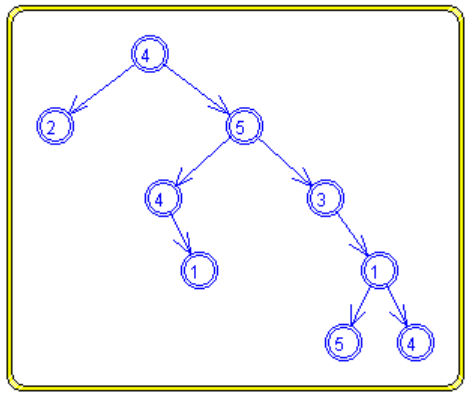
Joonis 16: Ülaltvaates kaetud tipud märgitud tärniga.



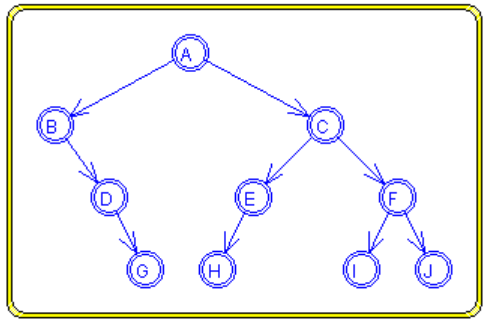
Joonis 17: Alumine saadud ülemisest ülesande 6.32 kohaselt.



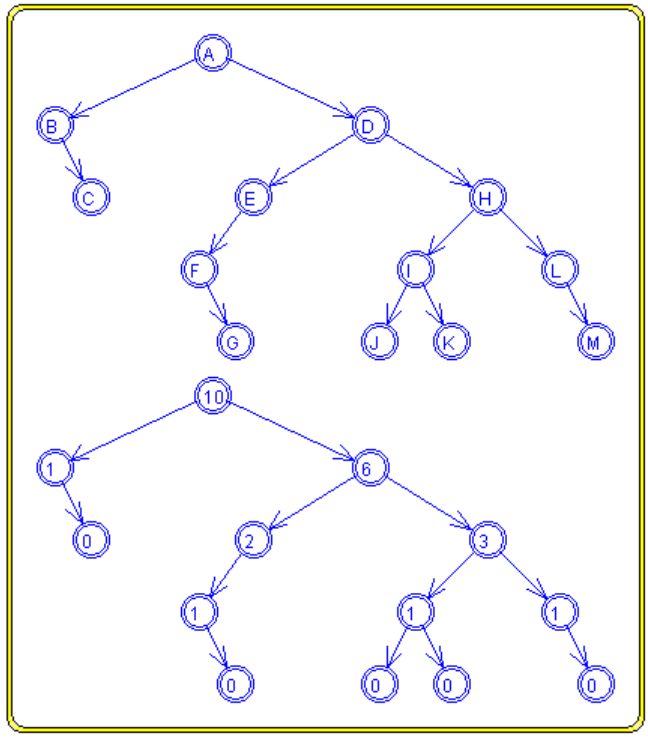
Joonis 18: Vasaku haru parempoolseim tipp on I sügavusega 3. Parema haru vasakpoolseim tipp on K sügavusega 1.



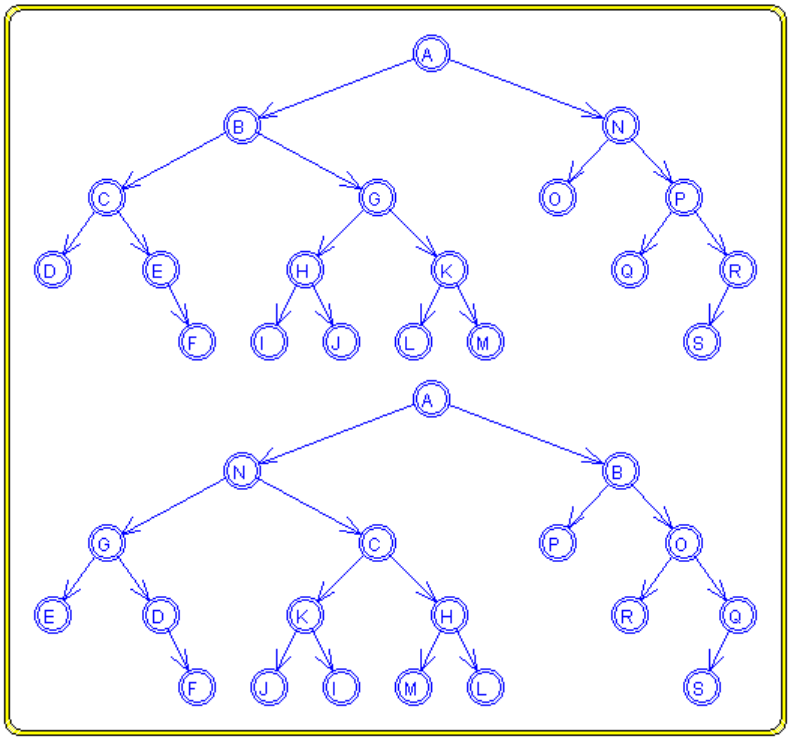
Joonis 19: Võtmete summa pikimal teel juur⇒leht on 18 $(= 4+5+3+1+5)$.



Joonis 20: Selles kahendpuus on 3 ühe alluvaga tippu.



Joonis 21: Ülal: antud kahendpuu. All: selle koopia, kus tipu võtmeks omistatud alampuude tipuarvudest suurem.



Joonis 22: Naabrite märgendid vahetatud ülesande 6.37 kohaselt.

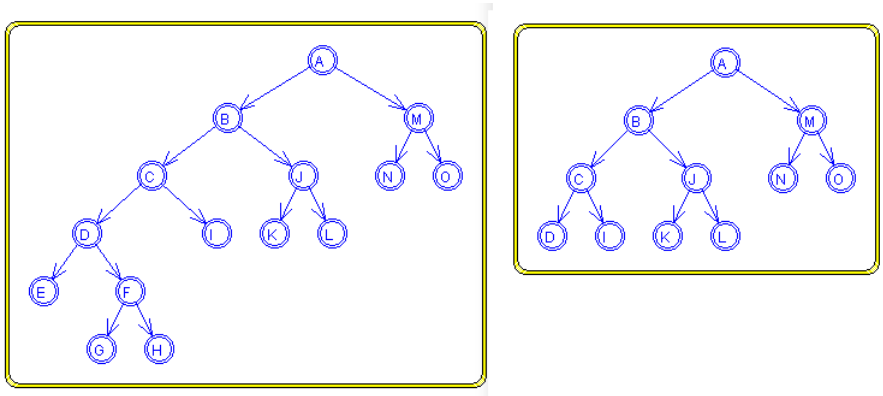
6.37. Luua kahendpuu, mis on saadud etteantud kahendpuust täistippude alluvate tipumärgendite vahetamisel. Vt näide joonisel 22.

6.38. Kontrollida, kas antud kahendpuus leidub tipp, millel on kaks ühe ja sama märgendiga alluvat.

6.39. Kahendpuu-kujulise struktuuriga trassil tehakse järjestikku katseid külastada lõpp- ehk lehttippe, liikudes ülemuselt alluvale, iga katset juurtipust alustades. Kahendpuu juurtipus ja igas vahetipus asub kahevalentne foor, milles põleb kas roheline või punane tuli. Roheline tähendab, et on lubatud liikuda ainult vasakule alluvale; punane foorituli lubab liikuda ainult paremale alluvale. Tipu läbimisel muutub foori tuli vastupidiseks: rohelisest punaseks või punasest roheliseks. Katse loetakse sooritatuks, kui jõutakse lehttippu või tippu, millel puudub see alluv, kuhu foor lubab liikuda; viimasel juhul foor ikkagi muudab värvi. Enne esimest katset on foorid lülitatud juhuslikul moel kas roheliseks või punaseks.

Induktsiooniga on tõestatav, et trassi mistahes tipp saab külastatud etteantud arv kordi lõpliku arvu katsetega.

Leida, mitu katset läheb tarvis, et kokkuvõttes oleks külastatud antud trassi iga lõpptipp.



Joonis 23: Antud täis kahendpuust (vasakul) tasemete kärpimisel saadud kompaktne kahendpuu (paremal).

6.40. Antud on täis kahendpuu. Kärpida minimaalne arv alumisi tasemeid, et tulemuseks saada kompaktne kahendpuu.

Vt näide joonisel 23.

6.41. Leida antud kahendpuu viimasel (sügavaimal) tasemel olevate tippude arv.

6.42. Leida antud tipu tasemenumber antud kahendpuus.

6.43. Luua etteantud kõrgusega täielik kahendpuu, milles tippude märgenditeks on tipunumbrid (nummerdamisel tasemeti, ülalt alla vasakult paremale).

6.44. Kontrollida, kas antud kahendpuu viimasel tasemel leidub täpselt üks tipp.

6.45. Antud on arvuliste tippudega kahendpuu ja arv k. Koostada antud kahendpuu koopia, seejuures paarisnumbriga tasemetel olevate tippude võtmetelele liita arv k. Vt näide joonisel 24, kus liidetav $k = 1$.

6.46. Antud on tasemenumber m. Loetleda antud kahendpuu m-ndal tasemel olevad tipud järjekorras paremalt vasakule. Vt näide joonisel 25.

6.47. Luua kahendpuu, mis saadakse antud kahendpuust etteantud arvust suurema sügavusega osade äralõikamisel. Vt näide joonisel 26.

6.48. Loetleda antud kahendpuu lehttipud tasemete kaupa, taseme sügavuse

(a) kasvamise järjekorras;

(b) kahanemise järjekorras.

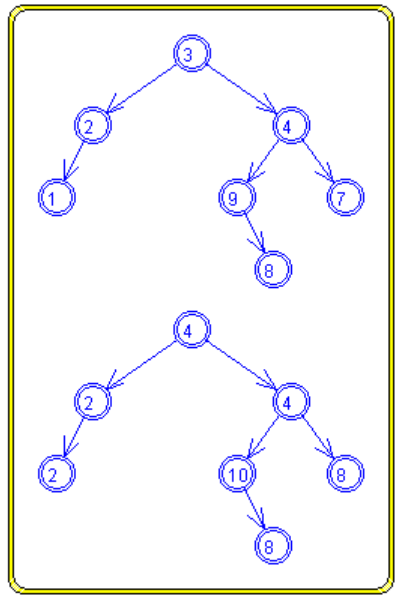
6.49. Antud kahendpuus leida lehttippude arv kõrgeimal lehttippe sisaldaval tasemel. Vt näide joonisel 27.

6.50. Kahendpuu *laiuseks* nimetatakse arvu $1+hv+hp$, kus hv on selle kahendpuu vasaku haru kõrgus ja hp – parema haru kõrgus.

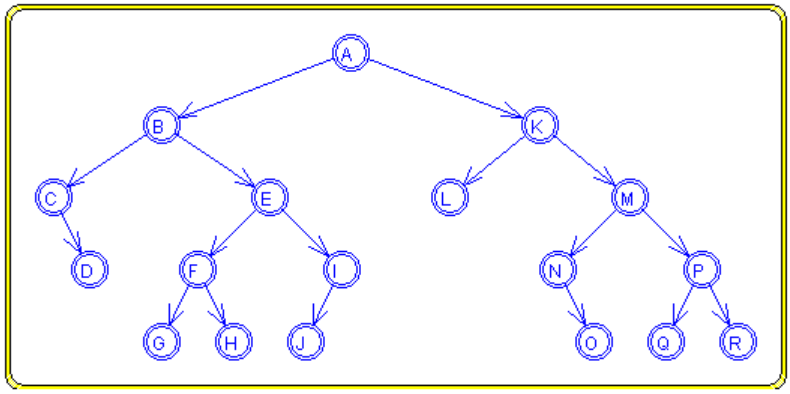
Kahendpuu *diameeter* on suurim selle kõikide alampuude laiustest.

Leida antud kahendpuu (üks) suurima laiusega (st diameetrit määrav) alampuu. Lisada tärn leitud alampuu juurtipu märgendile; samuti märgistada tärniga selle mõlemas harus ühe pikima tee tipud.

Vt näide joonisel 28.

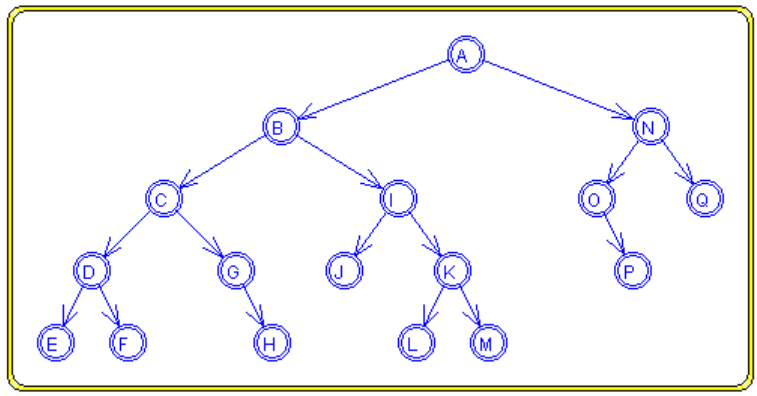


Joonis 24: Alumine kahendpuu on ülemise koopia, kuid arvudele tasemetel 0 ja 2 on liidetud 1.

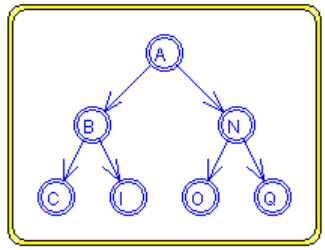


Tipud tasemel nr 3: P N I F D
Tipud tasemel nr 0: A

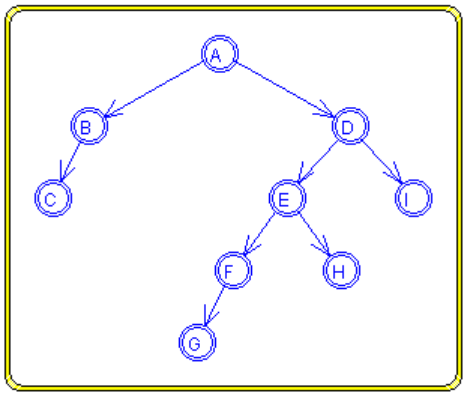
Joonis 25: Kahendpuu tippude loetelusid ülesande 6.46 kohaselt.



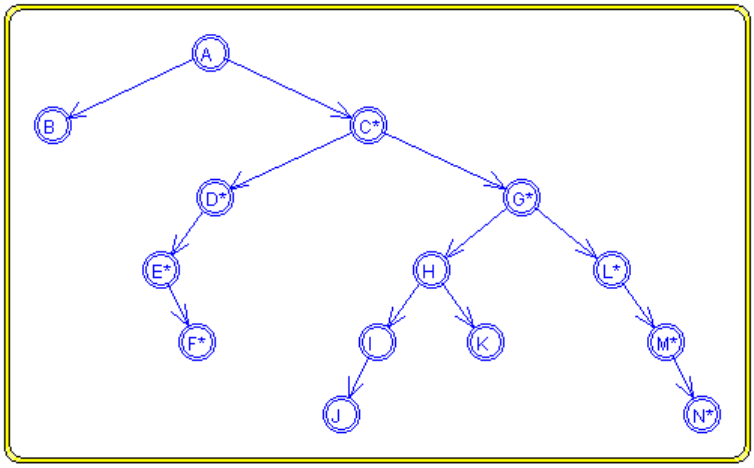
Kärbitud alates tasemest nr 3:



Joonis 26: Alumiste tasemete kärpimine.



Joonis 27: Kõrgeimal lehttippe sisaldaval tasemel on 2 lehttippu (C ja I).



Joonis 28: Kahendpuu diameetriga 8 (vt ka ülesanne 6.50).

7. Lisa: rekursioon

I Mõisteid

Rekursiivne stiil on programmeerimistehnika, kus sisendparameetri(te)ga määratud ülesanne taandatakse samasugus(t)ele, aga väiksema sisendparameetri(te) mahuga ülesandele (ülesannetele). Võimalik on ka olukord, kus rekursiivsel lahendusprotseduuril (või -funktsioonil) parameetreid pole, kuid selliseid me käesolevas ei käsitle.

Rekursiivne lähenemine on alternatiiviks paremini tuntud **iteratiivsele** stiilile, kus sellist väiksema mahuga alamülesannetele taandamist ei toimu, vaid ülesande lahendus kirjeldatakse tsükli(te) abil.

Triviaalsed ülesanded (nagu näiteks kahe arvu summa leidmine) on muidugi programmeeritavad ilma mingit spetsiifilist stiili järgimata.

7.1. Koostada Java-meetod

(a) massiivina antud järjendi,

(b) antud sõne

transpositsiooni leidmiseks. Hinnata koostatud meetodis kasutatava abimälu mahtu.

(Järjendi a transpositsioon: järjend a, milles elementide väärtuste järjestus on muudetud esialgsele vastupidiseks. Sõne s transpositsioon: sõne s', mis koosneb sõne s vastupidises järjestuses paigutatud sümbolitest.)

Selle ülesande (variant a) iteratiivse lahenduse näide:

```
static void transponeeridaIter(int[] a){// iteratiivne
// Antud:   arvujärjend massiivina a
// Tulemus: järjend a transponeeritud kohapeal,
//          st kasutades vaid tõkestatud mahuga lisamälu
// Idee: vahetada omavahel järjendi algusest ja lõpust
//       ühekaugel olevad arvud
   int i = 0, j = a.length-1; // vahetatavate indeksid
   while(i < j){
      // vahetada a[i] ja a[j]:
      int x = a[i];
      a[i]  = a[j];
      a[j]  = x;
      // muuta indeksid:
      i++; j--;
   }
}//transponeeridaIter
```

Abimälu moodustavad parajasti 3 lihtmuutujat, nimelt `i`, `j` ning `x`.

Teise ülesande (variant b) iteratiivse lahenduse näide:

```
static String transponeeridaIter(String s){// iteratiivne
// Antud:   sõne s
// Tulemus: tagastatakse sõne s transpositsioon (peegeldus)
// Rakendusega
//               s = transponeeridaIter(s);
// saab sõne s uueks väärtuseks argumendi s transpositsioon
   int n = s.length();
   char[] a = new char[n]; // abimassiiv
   for(int i = 0; i < n; i++)
      a[i] = s.charAt(n-1-i);
   return new String(a);
}//transponeeridaIter
```

Abimälu moodustavad siin kaks lihtmuutujat, n ja i, pluss abimassiivi n elementi.

Demonstreerime alljärgnevas rekursiivses stiilis lähenemist nõutud meetodite koostamisele ja toome esile seonduvaid mõisteid.

Mõisted: baasjuht, rekursiivne funktsioon, rekursiivne protseduur, sisendparameeter, sisend-väljundparameeter, rekursiivne protseduur-funktsioon

Olgu ülesandeks arvujärjendi transponeerimine (elementide järjekorra vastupidiseks muutmine). Üks võimalik rekursiivse lähenemise idee:

$$f(a) = [x, f(b), y] \tag{1}$$

kus x on a viimane element, y on a esimene element ja b on a ilma esimese ja viimase elemendita.

Näide:

$a = [3,5,2,4,7]$
$x = 7$
$y = 3$
$b = [5,2,4]$
$f[b] = [4,2,5]$
$f(a) = [7, f(b), 3] = [7,4,2,5,3]$

Paneme tähele, et $f(b)$ arvutatakse järgmiselt:

$b = [5,2,4]$
$x' = 4$
$y' = 5$
$b' = [2]$
$f(b) = [4, f[b'], 5]$

Siiski, $f(b')$ omakorda arvutamisel jääksime hätta:

$b' = [2]$
$x'' = 2$?
$y'' = 2$?
$b'' = []$?
$f(b'') = [??]$

See tähendab, et rekursiivse funktsiooni definitsioon (1) on puudulik. Vajalik on lisada nn **baasjuht** – funktsiooni väärtuse arvutamine juhul, kui parameetrina antud andmete maht on juba nii väike, et tulemus on saadav ilma järjekordse rekursiivse rakenduseta. Antud näites on selleks juht, kus transponeeritavas lähtejärjendis on vaid üks element; tulemuseks on siis antud järjend ilma muutusteta. Tegelikult ei vaja transponeerimist ka lausa tühi järjend (selleni jõutaks, kui lähtejärjendis oleks paarisarv elemente). Seega järjendi $a = (a_0, a_1, \ldots, a_{n-1})$ korrektne transponeerimise **rekursiivse funktsiooni** definitsioon on:

$$f(a) = \begin{cases} (a_{n-1}, f(b), a_0), \text{ kui } a \text{ elementide arv } n > 1, \\ a, \text{ vastasel korral} \end{cases}$$

kus $b = (a_1, \ldots, a_{n-2})$.

Kui spetsifitseeriksime transponeerimise funktsiooni Java-meetodina:

```
static void transp(int[] a)
// Antud: järjend  (massiivina) a -- lähtejärjend,
//         mis tuleb transponeerida
// Tulemus: antud järjend transponeeritud
//          (elementide järjekord muudetud vastupidiseks)
```

siis oleks andmemaht fikseeritud võrdseks järjendi pikkusega ja sama meetodit (kahe võrra) väiksema andmemahuga ei ole võimalik selles meetodis rekursiivselt rakendada.

Soovitava rekursiivse meetodi peab kirjeldama üldisemalt, jättes ka võimaluse andmemaht nö parametriseerida. Näiteks:

```
static void transponeeridaRek(int[] a, int i, int j){
// Antud:   arvujärjend massiivina a ja indeksid i, j sellel
// Tulemus: järjendi alamjärjend a[i..j] transponeeritud
// Idee:    vahetada omavahel a[i] ja a[j] väärtused,
//          seejärel transponeerida nende vahele jääv osa
// Rakendamine terve massiivi a transponeerimiseks:
//                   transponeerida(a, 0, a.length-1);
   // baasjuht:
   if(i >= j) // osas a[i..j] on vähem kui 2 elementi
      return; // midagi ei tule teha

   // vahetada a[i] ja a[j]:
   int x = a[i];
   a[i]  = a[j];
   a[j]  = x;

   transponeeridaRek(a, i+1, j-1);

}//transponeeridaRek
```

Siinkohal märgime, et rekursiivsete meetodite korral on nii mahulise keerukuse (abimälu vajaduse) kui ka ajalise keerukuse hindamine mõnevõrra raskem, vähem läbinähtav kui iteratiivsetel juhtudel. Näiteks, ülaltoodud meetod `transponeeridaRek` kasutab abimälu märksa rohkem kui ainult üks lihtmuutuja `x`: igal rekursiivsel pöördumisel võetakse kasutusele uus mälukoht selle muutuja jaoks (vabaneb alles sellest pöördumisest väljumisel). Mahulist keerukust vähendaks muutujast `x` loobumine – kui kahe elemendi väärtuse vahetamiseks kasutada eraldi meetodit, nt

`void vahetada(int[]a, int i, int j).`

Kahjuks kaasneks sellega omakorda transponeerimismeetodi töökiiruse mõningane vähenemine.

Paneme tähele, et rangelt võttes meetod `transponeeridaRek` ei ole funktsioon, kuna selle rakendamisel mingit väärtust ei tagastata. Rakendamise tulemuseks on parameetri `a` muudetud väärtus. Tulemuse leidmiseks kasutatakse `a` esialgset väärtust ning veel parameetrite `i` ja `j` väärtusi.

Taolist realisatsiooni on täpsem nimetada **rekursiivseks protseduuriks**. Viimase rakendamisel midagi ei tagastata; parameetriteks on peale **sisendparameetrite** (mille väärtusi kasutatakse argumentidena) veel üks või mitu **sisend-väljundparameetrit**, mille protseduuri sisenemisel antud väärtust kasutatakse argumendina ja millele protseduuri töö käigus omistatakse leitud (tulemuse) väärtus. Protseduuril võivad olla ka ainult sisendparameetrid, nt juhul kui tulemuseks on mingi tegevuse (nt printimise) sooritamine.

Rekursiivsel protseduuril `transponeeridaRek` on parajasti üks sisend-väljundparameeter, milleks on `a`.

Mõnel juhul võib olla tegemist ka nö sega-realisatsiooniga – **rekursiivse protseduur-funktsiooniga**, kus, lisaks sisend-väljundparameetri (uuele) väärtusele või soovitava tegevuse sooritamisele, mingi osa tulemusest moodustab käsuga `return` tagastatav väärtus.

Rekursiivse meetodi (funktsiooni) sõne transpositsiooni leidmiseks saab esitada erakordselt lakooniliselt:

```
static String transponeeridaRek(String s){// rekursiivne
// Antud:   sõne s
// Tulemus: tagastatakse sõne s transpositsioon (peegeldus)
// Rakendusega
//              s = transponeeridaRek(s);
//  saab sõne s uueks väärtuseks argumendi s transpositsioon

   // baasjuht:
   if(s.length() == 0) // antud on tühi sõne
      return "";       // tulemuseks on tühi sõne

   return transponeeridaRek(s.substring(1)) + s.charAt(0);

}//transponeeridaRek
```

Näide on õpetlik, sest tegemist on olukorraga, kus väliselt lihtsa rekusiivse meetodi töö aeg kasvab andmemahu (siin: transponeeritava sõne pikkuse) kasvamisel väga kiiresti. Olenevalt sõne pikkusest võib see meetod olla kümneid ja isegi sadu kordi aeglasem, võrreldes vastava (eespool toodud) iteratiivse meetodiga `transponeeridaIter`. Põhjus on selles, et rekursiivse töö käigus luuakse siin "ohjeldamatult" uusi sõnesid-vahetulemusi – uute rekursiivsete väljakutsete tarvis.

Mõisted: unaarne rekursioon, sabarekursioon, binaarne rekursioon

7.2. Programmeerida Java-meetod massiivina antud arvujärjendi elementide summa arvutamiseks, kui rekursiooni skeemiks on

(a)

$$f(a) = \begin{cases} 0, \text{ kui } a \text{ on tühi järjend,} \\ a_0 + f(b), \text{ vastasel korral} \end{cases}$$

kus b on a ilma esimese elemendita;

(b)

$$f(a) = \begin{cases} 0, \text{ kui } a \text{ on tühi järjend,} \\ a_0, \text{ kui } a \text{ on üheelemendiline järjend,} \\ f(b') + f(b''), \text{ kui järjendis } a \text{ on rohkem kui 1 element} \end{cases}$$

kus b' on järjendi a esimene pool ja b'' selle teine pool.

Ülesande 7.2(a) lahendus rekursiivse funktsioonina:

```
static double summa(double[] a, int i){
// Antud: järjend massiivina a ja üks indeks i sellel
// Tulemus: arvutatakse ja tagastatakse antud järjendi
//          elementide summa alates elemendist a[i] kuni a lõpuni

   // baasjuht:
   if(i == a.length) // liidetavaid ei ole
      return 0.0;    // 0 elemendi summa

   return a[i] + summa(a, i+1);

}//summa
```

Siin on tegemist **unaarse rekursiooniga** (ehk lineaarse rekursiooniga) – rekursiivne rakendamine on ette nähtud ainult üks kord funktsiooni (või protseduuri) kehas. Olukorda, kus see ainuke rekursiivne rakendamine toimub funktsiooni keha lõpus (viimase tegevusena) nimetatakse **sabarekursiooniks** (ingl k *tail-recursion*).

Järgnev Java-meetod kujutab endast järjendi summa leidmise rekursiivset p r o t s e d u u r i:

```
static void summa(double[] a, int i, double[] s){
// Antud: a - järjend massiivina
//        i - üks indeks järjendil
//        s - üheelemendiline list s, mille ainsa elemendi
//            s[0] väärtuseks on summa a[0] + ... + a[i-1]
// Tulemus:  s[0] = a[0] + ... + a[i-1] + a[i]
// Sisendparameetriteks on a ja i
// Sisend-väljundparameetriks on s
// Rakendamine:
//       double[] s = new double[1];
//       summa(a, 0, s);
//       ------- siin s[0] väärtuseks on a elementide summa

   if(i < a.length){
      s[0] += a[i++];
      summa(a, i, s);
   }
   // baasjuht: i >= a.length, siis ei tee midagi

}//summa
```

Märgime, et sabarekursiooniga protseduur on täitmiskäigult väga lähedane iteratiivsele, tsüklina esitatud tegevusele, antud juhul näiteks Java-meetodile

```
static double summa(double[] a){
// Antud:    järjend massiivina a
// Tulemus:  tagastatakse a elementide summa
   double s = 0.0; int i = 0;
   while(i < a.length)
      s += a[i++];
   return s;
}//summa
```

Kuid **binaarse rekursiooni** juhul, kus funktsiooni või protseduuri kehas on ette nähtud rekursiivne väljakutse kahel korral, ei pruugi (loomulikku) iteratiivset analoogi alati leidudagi.

Esitame selle jaotise lõpuks veel paar binaarse rekursiooni näidet.

Ülesande 7.2(b) lahendus rekursiivse funktsioonina:

```
static double summaBin(double[] a, int i, int j){
// Antud:   arvujärjend massiivina a ja indeksid i, j sellel
// Tulemus: tagastatakse alamjärjendi a[i..j] summa
// Idee:    järjendi a[i..j] poolitamine,
//          järjendi summa = selle poolte summade summa
   if(i > j)
      return 0;
```

```
   if(i == j)
      return a[i];
   int k = (i + j) / 2; // a keskmise elemendi indeks
   return summaBin(a, i, k) + summaBin(a, k+1, j);
}//summaBin
```

Esitame siinkohal veel ühe sabarekursiooniga Java-meetodi:

```
static int fibo(int n, int i, int a, int b){
// Antud: n -- otsitava Fibonacci arvu number, n != 0
//        i -- parajasti leitava Fibonacci arvu number
//        a -- eelmine leitud Fibonacci arv
//        b -- üle-eelmine leitud Fibonacci arv
// Tulemus: tagastatakse Fibonacci arv järjekorranumbriga n
// Rakendamise näide (printida Fibonacci arv nr 11):
//            System.out.println(fibo(11, 0, 0, 1));

   int c = a + b; // järgmine Fibonacci arv

   // baasjuht:
   if(i == n)  // jooksev nr = vajalik nr
      return c;

   return fibo(n, i+1, c, a);

}//fibo
```

Selles arvestatakse, et Fibonacci arvud moodustavad arvujada, kus kaks esimest (järjekorranumbritega 0 ja 1) on arvud 0 ja 1 ning iga järgmine arv on eelmise kahe arvu summa.

Mõiste: rekursioonipuu

7.3. Leida antud järjekorranumbriga Fibonacci arv.

Fibonacci jada defineeritakse rekurrentse seosega

$$F_n = F_{n-1} + F_{n-2}$$

algtingimustel $F_0 = 0$ ja $F_1 = 1$.

Ehk siis:

$$F_n = \begin{cases} 0, \text{ kui } n = 0, \\ 1, \text{ kui } n = 1, \\ F_{n-1} + F_{n-2}, \text{ kui } n > 1 \end{cases}$$

Üks võimalus (kuigi arvutuslikult ebaefektiivne) konkreetse F_n arvutamiseks on mõlema vajaliku liidetava, F_{n-1} ja F_{n-2} leidmiseks rakendada omakorda (st re-

kursiivselt) seda sama funktsiooni. Niimoodi toimiv, binaarse rekursiooniga Java-meetod näeb välja järgmine:

```
static int fibo(int n){
// Antud: n, n >= 0
// Tulemus: tagastatakse Fibonacci arv järjekorranumbriga n

   // baasjuht:
   if(n <= 1)   // n on 0 või 1
      return n;

   return fibo(n-1) + fibo(n-2);

}//fibo
```

Märgime, et tagastuskäsk teostatakse alles siis, kui on leitud väärtus avaldisele `fibo(n-1) + fibo(n-2)`. Selle väärtus aga leitakse järgmiste tööoperatsioonide järjest teostamisel:

1) täidetakse rekursiivne väljakutse `fibo(n-1)`;
2) täidetakse rekursiivne väljakutse `fibo(n-2)`;
3) liidetakse väljakutsetel saadud kaks arvu ja tagastatakse see summa `fibo(n)` väärtusena.

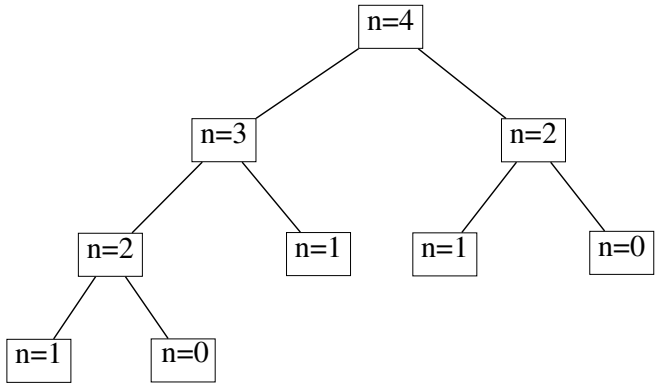
Rekursioonipuu konstrueeritakse mingi rekursiivse funktsiooni f (või rekursiivse protseduuri) jaoks, selgitamaks, mis järjekorras ja milliste sisendparameetri(te) väärtus(t)ega toimuvad funktsiooni (protseduuri) rekursiivsed väljakutsed selle konkreetse rakenduse, st konkreetsete sisendandmete korral.

Rekursioonipuu

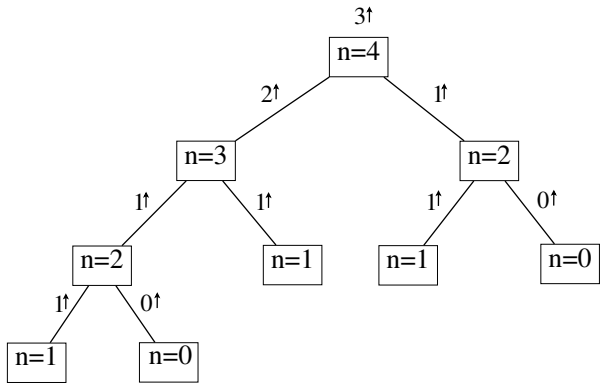
- tippudeks on funktsiooni/protseduuri väljakutsed fikseeritud parameetri(te)l; tipu märgendina näidatakse iga (sisend)parameetri nimi ja selle konkreetne väärtus;
- tipu t ülemuseks rekursioonipuus on tipp, millest on algatatud tipus t olev väljakutse;
- servad rekursioonipuu joonisel esindavad alluvussuhteid (nagu puude joonistel tavaks).

Funktsiooni `fibo` rekursioonipuu, kui konkreetselt arvutatakse `fibo(4)`, on kujutatud joonisel 29.

Rekursioonipuu lehttipuks on funktsiooni/protseduuri f rakendus baasjuhul. Binaarse rekursiooni korral on rekursioonipuuks kahendpuu, kus vasak alluv vastab esimesele, parem – teisele rekursiivsele väljakutsele. Kahendpuuna kujutatud väljakutsete jälgimine tehakse tavapärasel läbimise viisil: esmalt minnakse vasakusse harusse; kui terve vasak haru on läbitud, siis jätkatakse parema alluva ja selle vasaku haruga. Edaspidi, kui on tegemist rekursiivse funktsiooniga (mitte protseduuriga), siis saab puu servadel näidata ka väljakutse täitmise järel tagastatavaid funktsiooni väärtusi (suunaga alt üles), nagu nt joonisel 30.



Joonis 29: Funktsiooni `fibo(n)` rekursioonipuu argumendil $n = 4$.



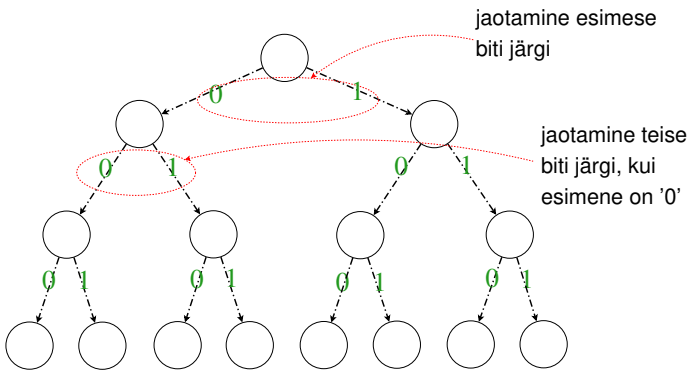
Joonis 30: Funktsiooni `fibo(n)` rekursioonipuu koos tagastatavate väärtustega.

7.4. Koostada üks unaarse ja üks binaarse rekursiooni skeem, kirjeldamaks järjendi elementidest vähima ja suurima väärtuse leidmist. Programmeerida nende põhjal kaks rekursiivset Java-meetodit.

II Binaarse rekursiooni rakendusi

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.

Järgnevalt kirjeldame näidete varal printsiipi, mis võimaldab samm-sammult konstrueerida etteantud ülesannet lahendavat rekursiivset programmi. 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 koosta-



Joonis 31: Võimaluste jaotamise skeem bitikolmikute korral.

da seda variantide osadeks jaotavat tööprotsessi (rekursioonipuud) järgivat rekursiivset programmi.

7.5. Koostada Java-meetod printimaks kõik n-bitised bitivektorid. Arv n antakse meetodi sisendparameetrina.

Lahendus. Kõikide variantide jaotusreegliks võiks olla see, et algul väljastame kõik need bitivektorid, mis algavad `'0'`-iga ja seejärel need, mis algavad `'1'`-ga.

Omakorda kõikide nende seas, kus esimene bitt on `'0'`, peame nende väljastamisel kehtestama mingi seaduspära. Näeme, et siingi sobib eelkirjeldatud reegel, et kõigepealt need, kus teine bitt on `'0'` ja seejärel need, kus teine bitt on `'1'`.

Kui tahame väljastada ekraanile kõiki selliseid bitijadasid, siis on loomulik seda teha lehttippudes. Seega vajame funktsiooni argumendiks juba „valitud“ bitte:

Joonisel 31 on kujutatud võimaluste jaotamine rekursioonipuus juhul $n = 3$.

Ülesannet lahendavas vastavas Java-meetodis lõpetame töö harus vaid siis, kui valitud bitte esitava järjendi pikkus on 3. Muudel juhtudel teostame rekursiivsed väljakutsed. Kokkuvõttes, kõigi bitikolmikute printimise meetod on järgmine:

```
static void bitiv3(String tee){
// Antud: tee -- senini leitud bitivektori osa (sõnena)
// Tulemus: prinditud kõik bitivektorid pikkusega 3
// Rakendamine: bitiv3("");
   // baasjuht:
   if(tee.length() == 3){ // tee on juba 3-bitine
      System.out.println(tee);
     return;
   }

   bitiv3(tee+'0'); // järgmisena lisame nulli
   bitiv3(tee+'1'); // järgmisena lisame ühe

}//bitiv3
```

Sellest saame hõlpsasti üldise ülesande (7.5) lahenduse:

```
static void bitiv(int n, String tee){
// Antud: n - nõutud bittide arv
//        tee -- senini leitud bitisõne
// Tulemus: prinditud kõik bitivektorid pikkusega n
// Idee: kui tee on n-bitiline, siis väljastud;
//       vastasel korral sooritatud see protseduur
//           0-ga pikendatud tee korral, seejärel
//           1-ga pikendatud tee korral
// Rakendamine: bitiv(n, "");

// baasjuht:
   if(tee.length() == n){ // tee on juba n-bitine
      System.out.println(tee);
      return;
   }

   bitiv(n, tee+'0'); // järgmisena lisame nulli
   bitiv(n, tee+'1'); // järgmisena lisame ühe

}//bitiv
```

Tegemist on rekursiivse protseduuriga, sest midagi ei tagastata.

7.6. Koostada Java-meetod printimaks kõik sellised 30-bitised bitivektorid, milles on täpselt 2 ühte (st `'1'`-bitti).

Vihje. Kontrollides `'1'`-bittide arvu alles 30-ndal tasemel kaotataks oluliselt töökiiruses.

7.7. Koostada Java-meetod, mis prindib järjendina antud hulga kõik alamhulgad.

Lahendus. Võimaluste jaotamine on siin sarnane kõigi bitivektorite genereerimise ülesandele. Näeme, et järjendi a kõik alamhulgad saame jaotada kaheks vastavalt sellele, kas element a_0 kuulub või ei kuulu valikusse. Nende mõlema jaotuse sees saame omakorda jaotada võimalusi vastavalt sellele, kas a_1 kuulub valikusse või mitte jne.

Seega koostatav funktsioon vajab kolme argumenti:

- `a` – vaadeldav järjend;
- `tee` – järjend valitud elementide salvestamiseks (väljastatakse lehttipus);
- `i` – tasemenumbri loendur järjendi elemendi indeksi fikseerimisel.

Rekursioonipuu juhul `a = [5,6,7], i = 0, tee = []` on (osaliselt) kujutatud joonisel 32. Vastav rekursiivne Java-meetod:

```
static void alamhulgad(int[] a, int i, List<Integer> tee){
// Antud: a - lähtehulk järjendina,
//        i - selle elemendi indeks järjendis a,
//            mis tuleks lisada alamhulgale tee
//        tee - jooksvalt täiendatav alamhulk listina
// Tulemus: prinditud kõik a alamhulgad
// Idee: prinditakse tee, kui sellele midagi lisada pole;
//       vastasel korral sooritatakse see protseduur
//       tee korral, seejärel a[i]-ga pikendatud tee korral,
//           andes mõlemal juhul uue,
//           vaadeldava elemendi indeksiks i+1

   // baasjuht:
   if(i == a.length){ // i on jõudnud a lõppu
      System.out.println(tee);
      return;
   }

   alamhulgad(a, i+1, tee); // ei võta a[i]-d

   // võtame a[i] (lisame selle listi tee koopia lõppu):
   tee = new ArrayList<Integer>(tee);
   // NB! tee on nüüd uus koopia parameetrist tee
   tee.add(a[i]);
   alamhulgad(a, i+1, tee);

}//alamhulgad
```

Rakendamise näide: käsuga

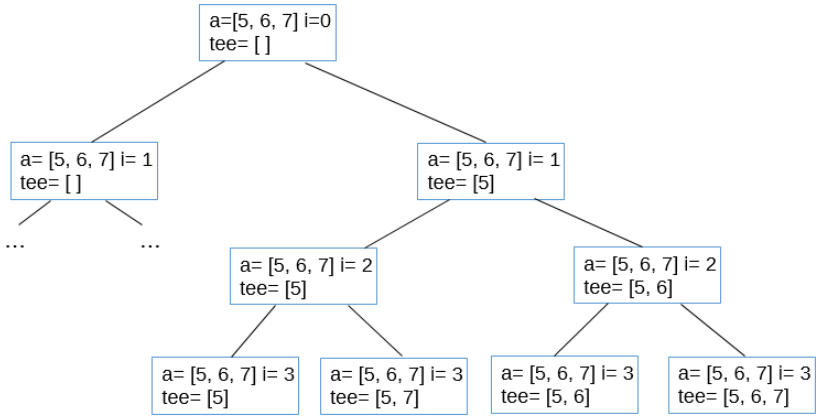
```
alamhulgad(new int[]{5,6,7}, 0, new ArrayList<Integer>());
```

prinditakse:

```
[]
[7]
[6]
[6, 7]
[5]
[5, 7]
[5, 6]
[5, 6, 7]
```

Binaarse rekursiooni skeemi alusel saab lahendada näiteks ka järgmised programmeerimisülesanded.

7.8. Koostada Java-meetod printimaks antud arvujärjendi kõik sellised osajärjendid, mille elementide summa jääb lõiku $[90, 100]$.



Joonis 32: Protseduuri `alamhulgad(a,i,tee)` rekursioonipuu (osaliselt).

7.9. Koostada Java-meetod printimaks antud arvujärjendi sellise osajärjendi elementide summa, mis ületab arvu 100 võimalikult vähe, olles parimal juhul sellega võrdne.

7.10. Koostada Java-meetod, mis leiab antud järjendi a pikima kasvava

(a) alamjärjendi;

(b) osajärjendi.

III Ternaarne rekursioon

Tegemist on juhuga, kus protseduuri või funktsiooni kehas nähakse ette kolm selle rekursiivset väljakutset.

7.11. Koostada rekursiivne Java-meetod, mis tagastab arvu, mitmel eri viisil saab üles minna n-astmelisest trepist, kui iga sammuga võib võtta ühe, kaks või kolm astet.

Lahendus 1. Selle ülesande lahendamise rekursiivne funktsioon $s(n)$ on disainitav järgmise ülalt-alla arutelu põhjal.

Olgu $n > 3$ trepi astmete arv. On päris selge, et viimasele astmele (nr n) jõutakse

(1) kas eelviimaselt (nr $n-1$) astmelt, võttes ühe astme;

(2) või eel-eelviimaselt (nr $n-2$) astmelt, võttes kaks astet;

(3) või eel-eel-eelviimaselt (nr $n-3$) astmelt, võttes kolm astet.

Rohkem võimalusi ei ole. Seega eri viiside arv, kuidas saab jõuda astmele nr n, avaldub kolme liidetava summana $x+y+z$, kus

(1) x on erinevate viiside arv, kuidas saab jõuda astmele nr $n-1$;

(2) y on erinevate viiside arv, kuidas saab jõuda astmele nr $n-2$;

(3) z on erinevate viiside arv, kuidas saab jõuda astmele nr $n-3$.

Liidetavad on aga arvutatavad rekursiivselt:

(1) $x = s(n-1)$;

(2) $y = s(n-2)$;

(3) $x = s(n-3)$.

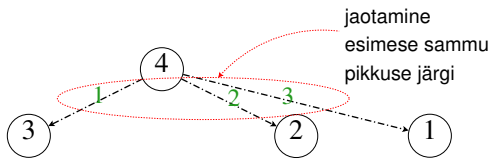
Baasjuhud, kui $n \leq 3$:
$s(1) = 1$ – ainult võttes ühe astme,
$s(2) = 2$ – võttes 2 ja 1, või 1 ja 2 astet,
$s(3) = 4$ – võttes astmeid $1,1,1$ või $1,2$ või $2,1$ või 3.
Arutelu tulemusena saame järgmise rekursiivse Java-meetodi:

```
static int s(int n){
// Antud:   trepi astmete arv n > 0
// Tulemus: tagastatakse arv, mitmel erineval moel saab üles
//          minna n-astmelisest trepist, kui iga sammuga
//          võib võtta ühe, kaks või kolm astet

   // baasjuht:
   switch(n){
     case 1:
        return 1;
     case 2:
        return 2;
     case 3:
        return 4;
   }

   return s(n-1) + s(n-2) +s(n-3);

}//s
```

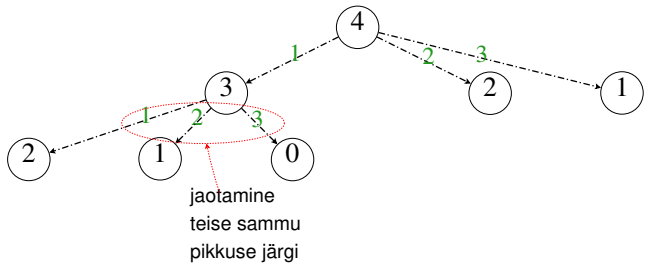
Lahendus 2. Skeemi saamiseks teeme läbi variantide läbivaatuse mingil parameetri väärtusel, olgu $n = 4$. Olemaks kindlad, et kõik ülesminemised saavad vaadeldud, jaotame need esimese sammu pikkuse järgi. See tähendab, et kõigepealt käsitleme need võimalused, kus esimese sammu pikkus on 1, seejärel võimalused, kus esimese sammu pikkus on 2 ja lõpuks võimalused, kus esimese sammu pikkus on 3 (joonis 33). Puu tipu märgendiks paneme veel astuda olevate trepiastmete arvu ja kaare märgendiks kasutatava sammu pikkuse.

Juhud, kus esimene samm on pikkusega 1, jaotuvad teise sammu järgi nagu joonisel 34. Jätkates edasi, saame täieliku võimaluste puutaolise skeemi (joonis 35).

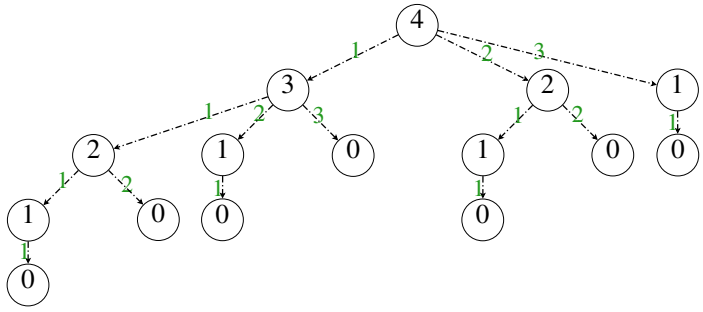
Näeme, et igas hargnemiskohas (skeemi tipus) käitume ühetaoliselt, harude arv on $1 \ldots 3$ ja see sõltub veel astumata astmete arvust.



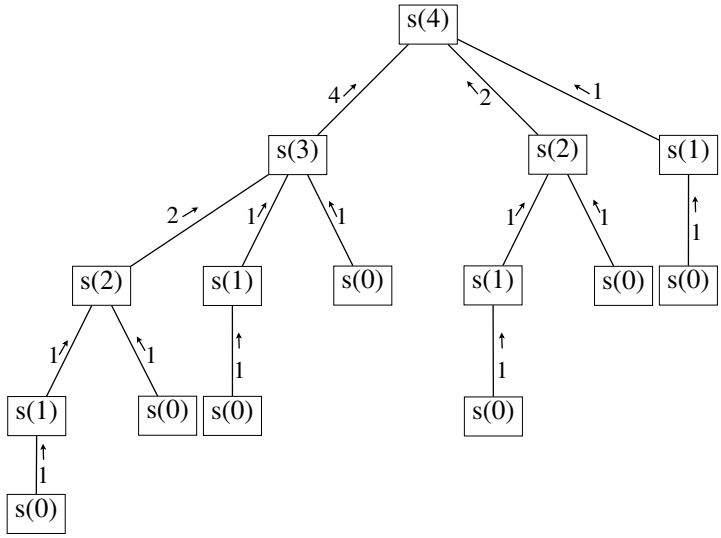
Joonis 33: Võimaluste jaotamine esimese sammu pikkuse järgi.



Joonis 34: Võimaluste jaotamine teise sammu pikkuse järgi (juhul, kui esimese sammu pikkus oli 1).



Joonis 35: Võimaluste jaotamise skeem trepist ülesminemisel.



Joonis 36: Väljakutse `s(4)` rekursioonipuu koos tagastatavate väärtustega.

Loodud skeemi järgi variante käsitledes peame vaid teadma, mitu trepiastet on veel üles astuda. Seega seda skeemi läbiva rekursiivse funktsiooni parameetriks on arv, mis iseloomustab, mitu astet on veel astuda.

Ülesande püstitusest lähtuvalt ei pea me haru lõppemisel midagi väljastama, meid huvitab vaid erinevate ülesminemiste arv. Seega loodav rekursiivne funktsioon peab väljakutsetel, mille täitmisel „töö lõpeb“, tagastama arvu 1; ülejäänud juhtudel tagastama harude „väärtuste“ summa (joonis 36).

Võttes baasjuhtudeks parameetri *n* väärtused 1, 2 ja 3, saame arutelu tulemusena samuti ülaltoodud rekursiivse Java-meetodi `s(int n)`.

Jooniste selguse huvides jätame edaspidi ära servadel asetsevad tagastamist iseloomustavad nooled.

Tähelepanek. Mitte just väga sportlik tavainimene korraga rohkem kui 2 astet ei võta. Ülalesitatud rekursiivse meetodi `s(n)` saab hõlpsasti kitsendada funktsiooniks

```
static int s_1_2(int n){
// Antud:   trepi astmete arv n > 0
// Tulemus: tagastatakse arv, mitmel erineval moel saab üles
//          minna n-astmelisest trepist, kui iga sammuga
//          võib võtta ühe või kaks astet

   // baasjuht:
   if(n <= 2) // n on 1 või 2
      return n;
   return s_1_2(n-1) + s_1_2(n-2);
}//s_1_2
```

Huvitav on see, et meetod `s_1_2(int n)` on peaaegu sama, mis Fibonacci arvu number n arvutamise rekursiivne meetod, vt ülesande 7.3 lahendus `fibo(int n)`. Ainult $s\_1\_2(n) = fibo(n+1)$, kui $n > 0$. Seega, seistes trepimarsi (ehk astmestiku) ees, kus on nt 11 astet, võib inimene kindel olla, et ülesmineku võimaluste arv (kui korraga võtta kas 1 või 2 astet) on F_{12}, mis on 144. Seega siin veel üks näide Fibonacci arvude esinemisest tavaelus.

7.12. Modifitseerida eelmise ülesande lahendusprogramm nii, et see prindiks ka kõikvõimalikud trepist ülesminemised. (Tulemuseks seega protseduurfunktsioon.)

Lahendus. Ilmselt on mõistlik ühel teekonnal kasutatud sammud väljastada siis, kui eeltoodud skeemi järgiv funktsioon on jõudnud lehttipuni. Selleks, et lehttipus oleks printimiseks vajalik info (nt sõnena sümbolitest `'1','2','3'`), peame sellel teekonnal seni astutud sammud argumendina kaasa viima. Seega on lahendusmeetodi signatuuriks

```
s(int n, String tee).
```

Printimine peab toimuma nüüd siis, kui parameeteri n väärtus on kas 1, 2 või 3. Kui

- $n = 1$, siis väljastada $tee +$ "1";
- $n = 2$, siis väljastada $tee +$ "11" ja $tee +$ "2";
- $n = 3$, siis väljastada $tee +$ "111", $tee +$ "12", $tee +$ "21" ja $tee +$ "3".

Ülejäänud juhtudel (võimalused jagunevad) tuleb kasutatud sammupikkus argumendina kaasa anda. Saame järgmise meetodi:

```
static int s(int n, String tee){
// Antud: n -- trepi astmete arv (n > 0),
//        tee -- kasutatud sammupikkuste järjend sõnena
// Tulemus: tagastatakse arv, mitmel erineval moel saab üles
//          minna n-astmelisest trepist, kui iga sammuga
//          võib võtta ühe, kaks või kolm astet;
//          ühtlasi prinditud kõik võimalikud sammupikkuste loetelud

   // baasjuhud:
   switch(n){
      case 1:
         System.out.println(tee + "1");
         return 1;
      case 2:
         System.out.println(tee + "11");
         System.out.println(tee + "2");
         return 2;
      case 3:
         System.out.println(tee + "111");
```

```
            System.out.println(tee + "12");
            System.out.println(tee + "21");
            System.out.println(tee + "3");
            return 4;
        }

        return s(n-1,tee+"1") + s(n-2,tee+"2") + s(n-3,tee+"3");

    }//s
```

Rakenduse näide:

```
    int n = 4;
    System.out.println("Trepist, millel on " + n + " astet, " +
                       "saab üles minna järgmiste sammudega: ");
    int m = s(n, "");
    System.out.println("-----------\nKokku " +
                                  m + " erinevat võimalust.");
```

Prinditakse:

```
Trepist, millel on 4 astet, saab üles minna järgmiste sammudega:
1111
112
121
13
211
22
31
-----------
Kokku 7 erinevat võimalust.
```

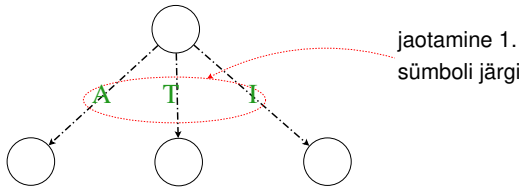
7.13. Modifitseerida ülalesitatud Java-meetod `s(n,tee)` nii, et

- see tagastaks kõikvõimalikud ülesminemised sõnede järjendina;
- selles on vaid 2 baasjuhtu.

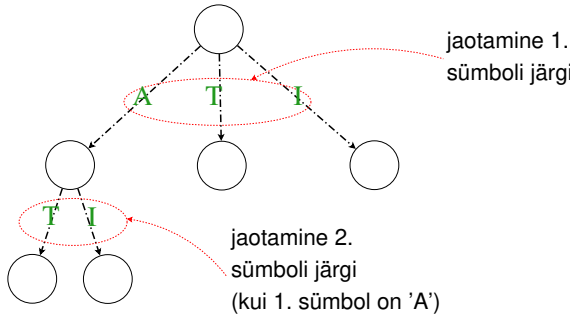
IV Multirekursioon

Multirekursiooni erijuhtudeks on ka juba vaadeldud binaarne ja ternaarne rekursioon. Mõnel keerukamal juhul aga tuleb rekursiivse funktsiooni või protseduuri kehas täitmisele muutuv arv (ka suurem kui 3) rekursiivseid väljakutseid, sõltuvalt konkreetsetest sisendparameetri(te) väärtustest.

Järgnevates näidetes 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.



Joonis 37: Permutatsioonide jaotamine esimese sümboli järgi.



Joonis 38: Permutatsioonide jaotamine teisel tasemel.

7.14. Kirjutada programm, mis väljastab ekraanile etteantud sõne kõik permutatsioonid.

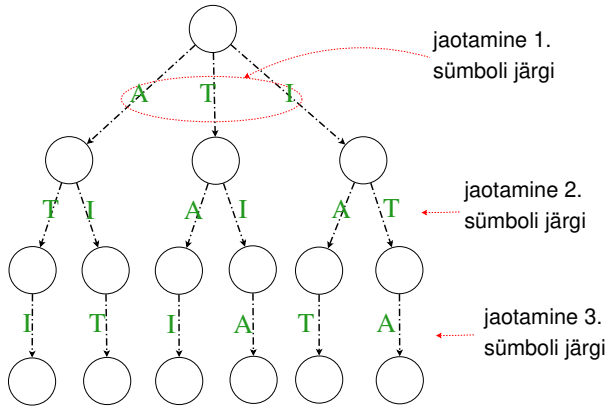
Lahendus. Selguse mõttes vaatleme variantide jaotamist konkreetse sisendi jaoks, olgu see `'ATI'`. Mõistlik variantide jaotamine osadeks võiks siis toimuda nii, et kõigepealt väljastame need järjestused, mis algavad sümboliga `'A'`, seejärel need, mis algavad sümboliga `'T'` ja lõpuks kõik sümboliga `'I'` algavad (joonis 37).

Juhtumid, kus esimene sümbol on `'A'`, saame jaotada vastavalt sellele, kas järgmine sümbol on `'T'` või `'I'` (joonis 38).

Edasi on igas tekkinud harus, ülesande parameetri väärtusest johtuvalt, vaid üks võimalus – valida üks sümbol üheelemendilisest hulgast. Sellega on näitesõna permutatsioonide hulk ammendunud (joonis 39).

Näeme, et jaotusskeemi igas harus toimime ühtemoodi – hargnemisel on harusid täpselt niipalju, kui on neid sümboleid, mida veel pole sellel teel „valitud“. Seega hargnemiste haldamiseks mingis tipus on vajalik teada, millised sümbolid on juba valitud. Kuigi valitutest saab tuletada, millised ei ole valitud, võtame ülevaatlikuse eesmärgil järgmises Python-programmis kasutusele ka sõne valimata sümbolitest.

```
static void perm(String s, String p){
// Antud:   s - permuteeritav sõne (millest veel valida)
//          p - järjekordne loomisel olev permutatsioon
// Tulemus: rakendusega
//                     perm(s, "");
```



Joonis 39: Sõne 'ATI' permutatsioonide jaotamise skeem.

```
//          prinditakse s sümbolite permutatsioonid

   if(s.length() == 0)
      System.out.println(p);
   else
      for(int i = 0; i < s.length(); i++){
         char c = s.charAt(i);
         perm(s.substring(0, i)+s.substring(i+1), p+c);
         //   --- s ilma i-nda sümbolita  c ---
      }
}//perm
```

Paljude ülesannete puhul tuleb variantide jaotamisprintsiip ise välja mõelda ja on üpris komplitseeritud. Järgneva ülesande lahendusega esitame veel ühe võimaliku jaotamisprintsiibi.

7.15. Koostada Java-meetod väljastamaks k kaupa kombinatsioonid antud järjendist.

Lahendus. Kombinatsioonide arvu n elemendist k kaupa esitab rekurrentne seos

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

algtingimustel $\binom{n}{0} = 1$ ja $\binom{n}{n} = 1$. Antud valemi järgi on küll selle arvu leidmine lihtne, kuid mitte kombinatsioonide endi leidmine.

Ühte võimalikku järjendi $a_0, a_1, \ldots, a_{n-1}$ k kaupa kombinatsiooni kirjeldab osajärjend $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$, kus $i_1 < i_2 < \ldots < i_k$. Kõik kombinatsioonid on siis võimalik jaotada ühisosata gruppideks vastavalt sellele, milline on selle osajärjendi esimese elemendi indeksi väärtus i_1.

Seega valime k kaupa kombinatsiooni esimese elemendi hulgast $\{a_0, a_1, \ldots, a_i\}$ ja ülejäänud $k-1$ elementi hulgast $a_{i+1}, \ldots, a_{n-1}$:

$$\underbrace{a_0, a_1, \ldots\ldots, a_{i-1}, a_i}_{\text{1. elemendi valik}} \underbrace{a_{i+1}, , \ldots\ldots, a_{n-1}}_{\text{valime } k-1 \text{ järgmist}}.$$

Alamjärjendist $a_{i+1}, \ldots, a_{n-1}$ peame valima ülejäänud $k-1$ elementi, selle võimaldamiseks peab siis

$$k-1 \leq n-1-(i+1)+1,$$

mida lihtsustades saame, et

$$i \leq n-k.$$

Seega jagunevad kõik k-kaupa kombinatsioonid vastavalt osajärjendi esimese elemendi indeksi väärtusele $i_1 = 0, \ldots, n-k$. Kirjeldatud jaotamisprintsiibile vastav kombinatsioone leidev Java-meetod on järgmine:

```
static void kombi(String s, String tee, int k){
// Antud: s - lähtesõne,
//        tee - hetkel väljavalitud sümbolid sõnena
//        k - mitu sümbolit vaja lisada juba valitutele (tee)
// Tulemus: rakendusega
//                     kombi(s, "", k);
//          prinditakse s sümbolite kombinatsioonid k kaupa

   if(k == 0){
      System.out.println(tee); // printida jrk kombinatsioon
      return;
   }

   int n = s.length();
   for(int i = 0; i < n-k+1; i++)
      kombi(s.substring(i+1), tee+s.charAt(i), k-1);

}//kombi
```

Selle protseduuri rekursioonipuus allapoole liikudes muutuvad argumendid järgnevalt.

- $s \mapsto s[i+1, ...]$ – järgmiste sümbolite valikut lubame hulgast $s_{i+1}, \ldots, s_{n-1}$;
- $tee \mapsto tee + s.charAt(i)$ – valitud sümboli s_i lisame tulemushulka;
- $k \mapsto k-1$ – vähendame veel valitavate arvu 1 võrra.

Meetodi näidiskäivituse

```
kombi("12345", "", 3);
```

väljundiks on

```
123
124
125
```

```
134
135
145
234
235
245
345
```

Kombinatsioone on tegelikult võimalik leida ka binaarse rekursiooniskeemi järgi.

7.16. Kirjutada meetod, mis prindib etteantud järjendist k-kaupa kombinatsioonid, kasutades binaarset rekursioonipuud.

Lisaülesanne. Võrrelda kahte kirjeldatud kombinatsioonide leidmise viisi. Kas need erinevad ajalise keerukuse mõttes?

V Generaatori mõiste

Eelnevates jaotistes vaatlesime rekursiivseid protseduure, mis väljastavad rea leitud väärtusi, nt bitivektoreid, alamhulki jmt. Praktikas on märksa suurema tähtsusega protseduurid, mis väljastamise (printimise) asemel genereerivad ehk annavad välja (*yield*) leitud väärtusi. Selliseid protseduure nimetatakse **generaatoriteks**.

Tutvustame siin Java generaatorklassi indeksite hulga $\{0,1,\ldots,n-1\}$ kombinatsioonide genereerimiseks. Tegemist on rekursiivse generaatoriga, kus iga indeksite k kaupa kombinatsioon saadakse mingist $k-1$ kaupa kombinatsioonist sellele ühe indeksi lisamise teel.

Näiteks, 3 kaupa kombinatsioon $(0,2,3)$ on saadud 2 kaupa kombinatsioonist $(0,2)$ indeksi 3 lisamisel, kombinatsiooni $(0,2)$ *laiendamisel*. Kombinatsioon $(0,2,4)$ aga saadakse kombinatsiooni $(0,2)$ laiendamisel indeksiga 4.

Kõik kombinatsioonid indeksitest $\{0,1,3,4\}$ kahe kaupa: (0, 1), (0, 2), (0, 3), (0, 4), (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4); kolme kaupa: (0, 1, 2), (0, 1, 3), (0, 1, 4), (0, 2, 3), (0, 2, 4), (0, 3, 4), (1, 2, 3), (1, 2, 4), (1, 3, 4), (2, 3, 4).

Märgime, et programmeerimiskeeles Java puudub jooksva väärtuse välja andmise käsk `yield`. Seetõttu on generaatorite programmeerimine Javas palju komplitseeritum kui nt Pythonis, kus selline käsk leidub.

Generaatorklass programmeeritakse Javas tavaliselt abstraktse liideseklassi, `Iterator` või `Iterable` realiseeringuna (*implements*). Loomulikult konkretiseeritakse seejuures abstraktsed meetodid `hasNext()` ja `next()`.

```
/*****************************************************
*              GENERAATOR-KLASS
*     indeksite hulga {0, 1, ..., n-1}
*     kombinatsioonide genereerimiseks.
*       Realiseerib liidese Iteraator
*
* Konstruktorile antakse
*  n -- mitmest indeksist ja k -- mitme kaupa.
* Isendimeetod next() annab listina välja järjekordse
*     kombinatsiooni hulgast {0, 1, ..., n-1} k kaupa.
*
* Seda klassi kasutatakse nagu iteraatorit ikka:
*   Gen_IndKombin gen = new Gen_IndKombin(n, k);
*   List<Integer> komb;
*   while(gen.hasNext()){
*      komb = gen.next(); // järjekordne kombinatsioon
*              ...
*   }//while
*
*****************************************************/

import java.util.Iterator;
import java.util.ArrayList;
import java.util.List;

class Gen_IndKombin implements Iterator<List<Integer>>{

   int n;        // mitmest
   int k;        // mitme kaupa
   long kombMax; // maksimaalne kombin. arv (n k)
   int loe;      // senini välja antud kombin. arv

   List<Integer> laiendatav = new ArrayList<Integer>();
   int laiendaja = 0;

   Gen_IndKombin gen; // koht generaatorile n-st k-1 kaupa

   /* Konstruktor
    * @param n: mitmest; n on
    *           vaadeldavate indeksite {0, 1, ..., n-1} arv.
    * @param k: mitme kaupa; k on arv hulgast {0, 1, ..., n}.
    */
   public Gen_IndKombin(int n, int k){
```

```
   this.n = n;
   this.k = k;
   kombMax = binomial(n, k);
   loe = (k == 0)? -1 : 0;

   if(k > 0)
      gen = new Gen_IndKombin(n, k-1); // rekursioon

}// konstruktor

@Override
public boolean hasNext(){
   return loe < kombMax;
}

@Override
public List<Integer> next(){
   if(k == 0){
      loe = 0;
      // anda välja tühi list:
      return new ArrayList<Integer>();
   }
L: for(;;){ // naasmispunkt
      // NB! Selle tsüklidirektiivi päisega märgistatakse
      //     koht programmis, kuhu saaks altpoolt naasta
      //     direktiiviga continue L;
      //     (markeerib Javas puuduvat käsku goto L)
      if(laiendaja < n){
         List<Integer> tulem =
                   new ArrayList<Integer>(laiendatav);
         tulem.add(laiendaja++);
         if(tulem.size() == k){
            // tulem on üks kombinatsioone k kaupa
            loe++;
            return new ArrayList<Integer>(tulem);
         }
         else
            continue L;
      }
      // siin: laiendaja == n
      if(gen.hasNext()){
         // järjekordne kombinatsioon k-1 kaupa:
         laiendatav = gen.next();
         laiendaja =
             laiendatav.get(laiendatav.size()-1) + 1;
```

```
            continue L;
         }
         else // k-1 kaupa kombinatsioone rohkem ei ole
            break L;

      }//for
      return null; // (kompilaatorile)
   }//next

   static long binomial(int n, int k){ // abimeetod
   // Antud: n, n > 0 ja k, 0 <= k <= n
   // Tulemus: tagastatakse kombinatsioonide arv n-st k kaupa
      if(k == 0)
         return 0;
      if(k == n)
         return 1;
      long r = Math.max(k, n-k)+1;
      for (long m = r+1, d = 2; m <= n; m++, d++ )
        r = r * m / d;
      return r;
   }//binomial

////////////////////////////////// TEST:
   public static void main(String[] args){
      for(int k = 0; k <= 5; k++){
         System.out.print("\nKombinatsioonid ");
         System.out.println("indeksitest 0, 1, 3, 4 ");
         System.out.println(k + " kaupa:");

         Gen_IndKombin gen = new Gen_IndKombin(5, k);
         List<Integer> komb;
         while(gen.hasNext()){
            komb = gen.next(); // järjekordne kombinatsioon
            System.out.println(komb);
         }//while
      }//for

   }//main

}//class
```

7.17. Koostada Java-meetod, mis tagastab antud täisarvude järjendi suurima elementide arvuga (ühe) osajärjendi, mille elementide summa on 0. Näiteks,
Antud: $(106,-83,138,89,-95,-53,119,47,133,134,-108,34,-72,1,132)$.
Tulemus: $(106,-83,89,-95,-53,47,134,-108,34,-72,1)$.

Lahendus. Genereerime indeksite $0, 1, \ldots, n-1$ kombinatsioone alaneva k kaupa, kus n on antud järjendi pikkus. Järjekordse osajärjendi saame, valides antud järjendi need elemendid, mille indeks on järjekordses kombinatsioonis. Lõpetame, kui järjekordse osajärjendi elementide summa on 0. Erijuhul, kui nullsummaga osajärjendit ei leidu, tagastame tühja järjendi.

```
static int[] nullsummaga(int[] a){
// Antud: täisarvujärjend a
// Tulemus: tagastatakse
//          järjendi a üks pikim nullsummaga osajärjend;
//          tühi järjend, kui sellist ei leidu
   int n = a.length;
   int[] tulem = new int[0];
   L: for(int k = n; k >= 0; k--){
         // vaadelda osajärjendeid pikkusega k
         Gen_IndKombin gen = new Gen_IndKombin(n, k);
         List<Integer> komb;
         while(gen.hasNext()){
            komb = gen.next(); // järjekordne kombinatsioon

            // summa nendelt a indeksitelt:
            int sum = 0;
            for(int i : komb)
               sum += a[i];

            if(sum == 0){ // kontroll
               // leitud, formeerida tulem:
               tulem = new int[k];
               int j = 0;
               for(int i : komb)
                  tulem[j++] = a[i];
               break L;
            }
         }//while
   }//for
   return tulem;
}//nullsummaga
```

Generaatorid on mõistlikuks alternatiiviks sellisele variantide töötlemisele, kus algul tehakse valmis kõikvõimalikud variandid ning alles seejärel toimub nende läbivaatus ja analüüs. Esiteks on generaatorid mälusäästlikud – kõiki variante pole tihtipeale vaja mälus hoida. Reeglina on variantide arv ka üsna suur, näiteks pikkusega n järjendi osajärjendite arv on 2^n. Teiseks, töökiiruse osas on generaatrid eriti efektiivsed siis, kui ülesande lahenduse saab paljudel juhtudel leida ilma kõiki variante läbi vaatamatagi.

7.18. Programmeerida permutatsioonide generaatorklass. Konstruktori sisendparameetriks on arv n, meetod `next()` tagastab järjekordse

(a) kordusteta (tavalise) permutatsiooni,

(b) kordustega permutatsiooni

indeksitest $\{0,1,\ldots,n-1\}$.

Kui n elemendi (indeksi) kordusteta permutatsioonide arv on teatavsti $n!$, siis kordustega permutatsioone on n^n.

Näiteks, juhul $n=3$ on indeksite $\{0,1,2\}$ kordusteta permutatsioone $3!=6$:
(0, 1, 2), (0, 2, 1), (1, 0, 2), (1, 2, 0), (2, 0, 1), (2, 1, 0),

kordustega permutatsioone aga $3^3=27$:
(0, 0, 0), (0, 0, 1), (0, 0, 2), (0, 1, 0), (0, 1, 1), (0, 1, 2), (0, 2, 0), (0, 2, 1),
(0, 2, 2), (1, 0, 0), (1, 0, 1), (1, 0, 2), (1, 1, 0), (1, 1, 1), (1, 1, 2), (1, 2, 0),
(1, 2, 1), (1, 2, 2), (2, 0, 0), (2, 0, 1), (2, 0, 2), (2, 1, 0), (2, 1, 1), (2, 1, 2),
(2, 2, 0), (2, 2, 1), (2, 2, 2).