1. Algoritmi ajalise keerukuse mõistest

Käesolev materjal on mõeldud mõningaseks täienduseks õpikule ([1], ptk 1).

Algoritmi ajaline keerukus on funktsioon f, mis igale selle algoritmi järgi lahendatavale konkreetsele ülesandele andmemahuga n seab vastavusse selle ülesande lahendamisel sooritatavate elementaaroperatsioonide arvu $f(n)$. Lihtsamatel juhtudel valitakse elementaaroperatsiooni rolli vaid üks „olulisim" algoritmis sooritatavatest tehetest, nn *põhitehe*. Näiteks faktoriaali arvutamise algoritmide korral on selleks loomulik valida korrutamistehe. Järgnevas on esitatud kaks faktoriaali leidmise algoritmi.

```
faktoriaal_1(n)
--- Antud: n, n >= 0
--- Tulemus: tagastatakse n!
        tulem := 1
        Iga i korral, i = 2, 3, ... ,n: tulem := tulem * i
   <=== tulem  --- tagastatakse muutuja tulem väärtus

faktoriaal_2(n)
--- Antud: n, n >= 0
--- Tulemus: tagastatakse n!
        Kui n < 2:
   <=== 1
        Vastasel korral:
   <=== faktoriaal_2(n-1) * n
```

On üsna ilmne, et nendel algoritmidel on täpselt sama ajaline keerukus (korrutamistehete arvu seisukohalt). Mõlema algoritmi korral avaldub ajaline keerukus (andmemahu n suhtes) kujul

$$f(n) = \begin{cases} 0 & \text{kui } n = 0 \\ n-1 & \text{kui } n > 0 \end{cases}$$

Andmekogumite (nt järjendite) töötlemise algoritmides sooritatavate elementaaroperatsioonide (loendatavate põhitehete) arv ei tarvitse sõltuda ainuüksi andmemahust (nt järjendi elementide arvust). Ühe ja sama mahuga (elementide arvuga) n andmekogumite töötlemisel tehtavate operatsioonide arv võib olla erinev, sõltuvalt andmete iseärasustest. Sel juhul ülaldefineeritud ajalise keerukuse mõiste ei ole kohaldatav, sest kujutis $\{n\} \to \{operatsioonide\ arv\}$ ei ole ühene, st ei ole funktsioon.

Näiteks algoritmi

```
sorteerituse_kontroll(a, n)
--- Antud: arvujärjend a ja selle elementide arv n
--- Tulemus: tagastatakse a mittekahanevalt sorteerituse
---          kontrollimise tulemus
    Iga i korral, i = 1, 2, ... , n-1:
           Kui a[i] > a[i+1]:
    <========== väär --- tagastatakse ja väljutakse
    <=== tõene --- tagastatakse (pärast tsükli töö lõppu)
```

rakendamisel 100-elemendiliste järjendite kontrollimiseks võib põhitehte – kahe elemendi võrdlemiste – arvuks olla nii 1 kui ka 99, üldiselt – iga täisarv lõigult $[1;99]$.

Niisuguste juhtudel sobib kasutada järgmist üldisemat, algoritmi keskmise ajalise keerukuse mõistet.

Algoritmi *keskmiseks ajaliseks keerukuseks* nimetatakse funktsiooni f, mis igale selle algoritmi sisendi andmemahule n seab vastavusse keskmise sellise andmemahuga ülesannete lahendamisel sooritatavate põhitehete arvu $f(n)$.

Ülaltoodud sorteerituse kontrollimise algoritmi korral on keskmiseks ajaliseks keerukuseks $f(n) = n/2$.

Vahel on otstarbekohane rakendada ka mõisteid *ajaline keerukuseks halvimal juhul* ja *ajaline keerukus parimal juhul*, mida me siinkohal ei käsitle. Mainime vaid, et sorteerituse kontrollimise algoritmile annavad kõige rohkem tööd (on halvimaks juhuks) sorteeritud järjendid; kõige kiiremini aga saab kontrollimisega ühele poole järjenditega, mille esimene element on suurem teisest elemendist (need sisendjärjendid moodustavad parima juhu).

Tihtipeale mõistetakse ajalise keerukuse all vaikimisi just keskmist ajalist keerukust.

Algoritmi ajaline keerukus iseloomustab algoritmi eeskätt selle töökiiruse (ajalise efektiivsuse) seisukohalt. Erinevate algoritmide efektiivsuse võrdlemisel ja klassifitseerimisel on aluseks nende ajalise keerukuse asümptootilised hinnangud ([1], ptk 1) - ajalise keerukuse „käitumine“ algandmete mahu (ajalise keerukuse argumendi) piiramatul kasvamisel. Ülalvaadeldud algoritmid faktoriaali arvutamiseks ja järjendis sorteerituse kontrollimiseks on lineaarse ajalise keerukusega (ajaline keerukus on lineaarfunktsioon). Lihtsamad sorteerimisalgoritmid on aga ruutkeerukusega (ajaline keerukus avaldub ruutfunktsioonina); tavaliselt loetakse nende korral põhitehteks, mille arvu loendatakse, järjendi kahe elemendi (elementide-kirjete korral – võtmete) võrdlemine. Algoritmide ajalise keerukuse asümptootilise hindamise ja vastavalt nende keerukusklassidesse jaotumise küsimused on juba omaette teema.

I Testülesanded

1.1. *(v)* Faktoriaali arvutamisel (algoritmi `faktoriaal_1` või `faktoriaal_2` järgi) saadi vastuseks arv 2004189184. Järelikult sisendparameetri n väärtuseks oli arv ..... , sooritati ..... korrutamistehet.

1.2. *(v)* Näidata, et arvu m^2 faktoriaali arvutamisel tehtavate korrutamistehete arvuks on $(m+1)(m-1)$.

2. Väga kiiresti kasvava ajalise keerukusega algoritmidest

Järgnevas vaatleme paari keerukamat, variantide läbivaatamisega seotud ülesannet. Üldiseks eesmärgiks on leida seise $n \times n$ malelaual, kuhu on paigutatud teatav arv ratsusid nii, et ükski ratsu ei ole ühegi teise ratsu tules (ratsude värvi arvestamata). Vaadeldavate ülesannete jaoks käesolevas koostatud algoritmide ajalised keerukused on väga kiiresti kasvavad funktsioonid. Nende asümptootilised hinnangud on siin väheolulised, sest argumendi n piiramatu „kasvatamine“ ei ole sisukas – praktiliselt võib piirduda juhtudega $1 \leq n \leq 10$; erilist huvi pakuks muidugi juht $n = 8$, st tavamõõtmetes malelaud.

Ülesanne 1. Antud täisarvude n ($n \geq 1$) ja r ($0 \leq r \leq \lfloor (n^2+1)/2 \rfloor$ korral leida kõikvõimalikud seisud $n \times n$ malelaual, kuhu on paigutatud r ratsut „rahumeelselt“, st nii, et ükski ratsu ei ole ühegi teise ratsu tules. ($\lfloor x \rfloor$ on arvu x alumine täisosa.) Näiteks $n = 4$ ja $r = 8$ korral on vastuseks seise kujutavad bitimaatriksid

```
1 0 1 1   1 0 0 1   1 0 1 0   1 1 1 1   0 1 0 1   1 1 0 1
0 0 0 1   1 0 0 1   0 1 0 1   0 0 0 0   1 0 1 0   1 0 0 0
1 0 0 0   1 0 0 1   1 0 1 0   0 0 0 0   0 1 0 1   0 0 0 1
1 1 0 1   1 0 0 1   0 1 0 1   1 1 1 1   1 0 1 0   1 0 1 1
```

kus 1 tähistab ruutu, millel seisab ratsu.

Sisendparameetri r ülemine tõke tuleneb asjaolust, et üldjuhul (kui $n \neq 2$) on $r_{max} = \lfloor (n^2+1)/2 \rfloor$ suurim võimalik arv rahumeelseid ratsusid $n \times n$ malelaual. Sellest suuremate sisendväärtuste r puhul oleks vastuseks seisude tühihulk.

Algoritm R0. Ülesande 1 lahendamine.

```
b := (mingi) bitijärjend pikkusega n*n, milles on r ühte
Järjendi b iga permutatsiooni p korral:
        M := maatriks(p); --- p teisendatakse bitimaatriksiks
        Kui sobib(M):
                väljastada(M)
```

Alamprotseduurideks on *maatriks*() ja *sobib*().

Teisendamisprotseduuri *maatriks*(p) spetsifikatsioon:

Antud: bitijärjend p, mille pikkus m on täisruut (1 või 4 või 9 või 16 jne)

Tulemus: tagastatakse bitimaatriks, mille järjekordseks reaks on p järjekordne alamjärjend pikkusega $\sqrt{m}$

Näiteks, kui p = [1 1 0 1 1 0 0 0 0 0 0 1 1 0 1 1] (m = 16), siis

maatriks(p) =

```
1 1 0 1
1 0 0 0
0 0 0 1
1 0 1 1
```

Kontrollimisprotseduuri *sobib*(M) spetsifikatsioon:

Antud: seis biti-ruutmaatriksina M, kus 1 tähistab ruutu, millel seisab ratsu

Tulemus: *tõene*, kui seisus M ükski ratsu ei ole ühegi teise ratsu tules, *väär* vastasel korral

Edasises loeme vaadeldavates algoritmides põhitehteks just kontrollimisprotseduuri *sobib* rakendamise.

Algoritmi R0 ajaline keerukus on ühemuutuja funktsioon $f0(n)$, mis igale n väärtusele seab vastavusse protseduuri *sobib* rakenduste arvu. Selles algoritmis rakendatakse kontrollimisprotseduuri järjendi b (pikkusega n^2) iga permutatsiooni p korral, seega *sobib* rakenduste arv $f0(n) = (n^2)!$. Antud juhul ajaline keerukus, st põhitehte *sobib*() arv, ei sõltu sellest, milline on antud r väärtus. Kuid leitud sobivate seisude arv on siiski sõltuv paigutatavate ratsude arvust r.

n	r	$f0(n) = (n^2)!$	Leitud seise
1	1	1	1
1	0	1	1
2	2	24	6
2	1	24	4
2	0	24	1
3	5	362880	2
3	4	362880	18
3	3	362880	36
...	...	...	...
4	8	2004189184	-
...	...	...	...

Tabeli viimases veerus on esitatud algoritmi R0 realiseeriva programmi tulemused. Nagu näha, algoritmi R0 ajaline keerukus $f0$ on väga kiiresti kasvav funktsioon. Algoritm ei ole praktiliselt kasutatav juba kolmest suurema n korral.

Algoritm R1. Ülesande 1 lahendamine.

```
Iga pikkusega n*n bitijärjendi b korral, milles on r ühte:
   M := maatriks(b); --- b teisendatakse bitimaatriksiks
         Kui sobib(M):
               väljastada(M)
```

Pikkusega m bitijärjendite arv, milles igaühes leidub parajasti k ühte, on C_m^k, st kombinatsioonide arv m elemendist k kaupa.

Algoritmi R1 ajaline keerukus on kahemuutuja funktsioon $f1(n,r)$, mis igale väärtuste paarile (n,r) seab vastavusse protseduuri *sobib*() rakenduste arvu. Selles algoritmis rakendatakse kontrollimisprotseduuri iga bitijärjendi b (pikkus n^2, parajasti r ühte) korral, seega *sobib*() rakenduste arv ehk algoritmi ajaline keerukus $f1(n,r) = C_{n^2}^r$. Selle funktsiooni mõningaid väärtusi on esitatud järgmises tabelis.

n	r	$f1(n,r) = C^r_{n^2}$	Leitud seise
1	1	1	1
1	0	1	1
2	2	6	6
2	1	4	4
2	0	1	1
3	5	126	2
3	4	126	18
3	3	84	36
3	2	36	28
3	1	9	9
3	0	1	1
4	8	12870	6
4	7	11440	48
...	...	...	...
6	18	9075135300	-
...	...	...	...
6	2	630	550
...	...	...	...

Algoritmi R1 ajaline keerukus $f1$ on üldiselt üsna kiiresti (kuigi mitte monotoonselt) kasvav funktsioon. Soodsamad on nähtavasti juhud, kus r on väike.

Ülesanne 2. Antud arvude $n(n \geq 1)$ ja $r(0 \leq r \leq \lfloor (n^2+1)/2 \rfloor$ korral leida kõikvõimalikud seisud $n \times n$ malelaual, kuhu on paigutatud vähemalt r (st r või rohkem) ratsut nii, et ükski ratsu ei ole ühegi teise ratsu tules.

Näiteks n = 3 ja r = 4 korral on vastuseks seise kujutavad 20 bitimaatriksit:

```
001  000  001  000  011  010  000  000  010  010
011  011  010  010  011  011  111  110  101  111
001  011  101  111  000  010  010  110  010  000

010  010  101  101  100  101  101  111  100  110
110  111  000  010  010  010  010  010  110  110
010  010  101  001  101  100  101  000  100  000
```

Algoritm R2. Ülesande 2 lahendamine (kui n > 3 on paarisarv).

```
R2(n, r)
--- Antud: n ja r  (kui n > 3, siis n on paarisarv)
--- Tulemus: tagastatakse kõigi selliste seisude
---          (nxn bitimaatriksite) hulk,
---           milles paikneb vähemalt r (st r või rohkem)
---           rahumeelset ratsut
 Baasjuhud (nt kui n <= 2). Tagastatakse "valmis" seisude hulk.
 tulem := tühi hulk --- tulemushulk
 m := täisosa(n/2)
 S := R2(m, 0); --- hulk S: ratsude kõik rahumeelsed seisud
 ---                        mxm laual
 A := (nullelementidega nxn maatriks)
 --- A veerandid (mxm maatriksid):
 ---     A1 A3
 ---     A2 A4
 Iga seisu s1 korral listist S:
   A1 <== s1; --- s1 panna A esimesse veerandisse
   Iga seisu s2 korral listist S:
      A2 <== s2; --- s2 panna A teise veerandisse
          Iga seisu s3 korral listist S:
             A3 <== s3; --- s3 panna A 3. veerandisse
             Iga seisu s4 korral listist S:
                A4 <== s4; --- s4 panna A 4. veerandisse
                   Kui sobib(A)
                     tulem <== A; --- A lisada hulka tulem
<=== tulem
```

Malelaud (A), mille mõõt (ridade arv) n on paaris, koosneb neljast $m \times m$ veerandist $(A1, A2, A3, A4)$, kus $m = n/2$.

Ilmselt $sobib(A) \Rightarrow sobib(A1) \wedge sobib(A2) \wedge sobib(A3) \wedge sobib(A4)$.

Algoritmis R2, selleks et leida r (vähemalt) ratsu rahumeelsed paigutused $n \times n$ lauale, leitakse (rekursiivselt) kõik seisud S, kus $m \times m$ lauale on paigutatud 0 või enam ratsut. Iga neljakaupa variatsiooni $(A1, A2, A3, A4)$ korral listist S komplekteeritakse neist veeranditest maatriks A ja see lisatakse tulemushulka *tulem*, juhul kui $sobib(A)$ on tõene.

Algoritmi R2 ajaline keerukus $f2$ on väga kiiresti kasvav, $f2(n,r) = k^4$, kus $k = |S| = R2(m,0)$ on ratsude kõikvõimalike rahumeelsete seisude arv $m \times m$ laual. Näiteks $R2(4,0) = 1365$, seega $f2(8,31) = 1365^4 = 3471607400625$. Järelikult selle algoritmiga ei saa praktiliselt leida vähemalt 31 ratsu kõiki rahumeelseid paigutusi 8×8 malelaual.

Ajalist keerukust saab siiski redutseerida järgmiselt.

I. Niipea, kui on sooritatud

```
A2 <== s2 (või  A3 <== s3)
```

kohe kontrollida sobib(A), ja mittesobivuse korral enam alamtsüklit

```
Iga seisu s3 korral listist S:
( või Iga seisu s4 korral listist S:)
```

mitte alustada.

II. Kaaluda võimalust: selmet leida S kui ratsude k õ i g i ($r = 0$) rahumeelsete seisude list $m \times m$ laua korral, kasutada v ä h e m a l t $r > 0$ ratsu rahumeelsete seisude hulka $m \times m$ laua korral. Paneme tähele, et järjekordses seisus A on ratsude arvuks $ratsudeArv(A) = a_1 + a_2 + a_3 + a_4$, kus $a_i = ratsudeArv(A_i), i = 1,2,3,4$. Veerandis A_i saab olla 0 kuni $aMax = \lfloor (m*m+1)/2 \rfloor$ ratsut, st $0 \le a_i \le aMax$. Igale leitud seisule A vastab liidetavate list $[a_1, a_2, a_3, a_4]$. Nende listide hulga saab koostada enne tsüklitega alustamist.

Näiteks, kui $n = 8$ ja $r = 30$ (ülesandeks vähemalt 30 ratsu kõik paigutused 8×8 malelaual), siis $ratsudeArv(A) \in \{30,31,32\}, aMax = 8$ ja summa 30 või 31 või 32 saamiseks on kasutada liidetavad $0,1,...,8$. Osutub, et liidetavana saab siin kasutada ainult arve 8, 7 ja 6 (arve $0,1,2,3,4,5$ aga mitte):

[8,8,8,8] summa 32
[7,8,8,8] summa 31
[8,7,8,8] summa 31
[8,8,7,8] summa 31
[8,8,8,7] summa 31
[6,8,8,8] summa 30
[7,7,8,8] summa 30
[7,8,7,8] summa 30
[7,8,8,7] summa 30
[8,6,8,8] summa 30
[8,7,7,8] summa 30
[8,7,8,7] summa 30
[8,8,6,8] summa 30
[8,8,7,7] summa 30
[8,8,8,6] summa 30

Seega keerukuse teatava redutseerimise tagab algoritmis R2 omistamise

```
S := R2(m, 0); --- hulk S: ratsude kõik rahumeelsed seisud
---                        mxm laual
```

asendamine omistamisega

```
S := R2(m, v); --- hulk S: kõikvõimalikud vähemalt v ratsu
---                rahumeelsed seisud mxm laual,
---                kus v on vähim liidetavatest,
---                mille nelikute summa >= r .
```

Selliselt muudetud R2 on juba praktiliselt rakendatav, näiteks vähemalt 31 ratsu kõikvõimalike rahumeelsete paigutuste leidmiseks 8×8 malelaual:

$|S| = |R2(4,7)| = 54$
$f2(8,31) = 54^4 = 8503056$
Vastava programmiga saab leida need 70 paigutust ($|R2(8,31)| = 70$).

Võimalik on ka leida vähemalt 30 ratsu kõikvõimalikud rahumeelsed paigutused 8×8 malelaual:

$|S| = |R2(4,6)| = 224$

$f2(8,30) = 224^4 = 2517630976$

Vastava programmiga on veel praktiliselt leitavad need $|R2(8,30)| = 1192$ paigutust.

Veel üks (kolmas) võimalus redutseerimiseks tuleneb asjaolust, et juhul $n > 4$, $r = \lfloor (n^2+1)/2 \rfloor$ on vastuseks ratsude paigutus(ed) üle ühe ruudu - kui n on paarisarv, siis parajasti kaks seisu:

10101 ...		01010 ...
01010 ...		10101 ...
...	ja	...
01010 ...		10101 ...

kui n on paaritu, siis parajasti üks seis:

10101 ...
01010 ...
...
10101 ...

See annab võimaluse kiiresti leida mõned $\lfloor (n^2+1)/2 \rfloor - 1$ ratsu rahumeelsed paigutused $n \times n$ malelaual. Järjekordse sellise seisu saab, asendades ülaltoodud maatriksis järjekordse biti 1 väärtusega 0. Paraku ei saada sellisel viisil kõiki soovitud paigutusi.

Ülesande 2 lahendamise algoritm, kui $n \geq 3$ on paaritu arv, on üsna analoogiline algoritmiga R2. Ainult A komplekteerimisel neljast veerandist asetatakse („kleebitakse“) veerandid üherealise/-veerulise ülekattega.

Malelaua A, mille mõõt (ridade arv) n on paaritu, komplekteerimine neljast ülekattega veerandist

$A1$ $A3$
$A2$ $A4$

toimub järgmiselt:

$A :=$ nullidest koosnev maatriks

$A1 \Rightarrow A$ esimesse veerandisse

$A2 \Rightarrow A$ teise veerandisse; ülekate $A1$ viimane rida, $A2$ esimene rida

$A3 \Rightarrow A$ kolmandasse veerandisse; ülekate $A1$ viimane veerg, $A3$ esimene veerg

$A4 \Rightarrow A$ neljandasse veerandisse; ülekate $A2$ viimane veerg, $A4$ esimene veerg ja ülekate $A3$ viimane rida, $A4$ esimene rida

Ülekaetavad bitiread/-veerud peavad olema sama konfiguratsiooniga, vastasel korral komplekteerida ei saa ning see juht tulemusesse midagi ei anna.

I Testülesanded

2.1. *(v)* Mitu seisu leidub 2×2 malelaua korral, milles 0 või enam ratsut on paigutatud nii, et ükski ratsu ei ole ühegi teise ratsu tules.

2.2. *(v)* Mitu seisu leidub 8×8 malelaua korral, milles parajasti 30 ratsut on paigutatud nii, et ükski ratsu ei ole ühegi teise ratsu tules.

2.3. *(v)* Leida algoritmi R1 ajalise keerukuse $f1$ väärtused
$f1(4,6) = ...$
$f1(4,5) = ...$
$f1(4,4) = ...$
$f1(4,3) = ...$
$f1(4,2) = ...$
$f1(4,1) = ...$
$f1(4,0) = ...$
$f1(5,2) = ...$
$f1(5,1) = ...$
$f1(5,0) = ...$

2.4. *(v)* Olgu S kõigi seisude hulk 6×6 malelaual, milles vähemalt 17 (17 või enam) ratsut paiknevad rahumeelselt. Iga seis $A \in S$ koosneb neljast 3×3 veerandist A_1, A_2, A_3, A_4. Seisus A olevate ratsude arv avaldub summana $ratsudeArv(A) = a_1 + a_2 + a_3 + a_4$, kus $a_i = ratsudeArv(A_i), i = 1,2,3,4$. Millised arvud saavad olla liidetavaks a_i $(i = 1,2,3,4)$?

2.5. *(v)* Olgu S kõigi seisude hulk 10×10 malelaual, milles vähemalt 48 (48 või enam) ratsut paiknevad rahumeelselt. Iga seis $A \in S$ koosneb neljast 5×5 veerandist A_1, A_2, A_3, A_4. Seisus A olevate ratsude arv avaldub summana $ratsudeArv(A) = a_1 + a_2 + a_3 + a_4$, kus $a_i = ratsudeArv(A_i), i = 1,2,3,4$. Millised arvud saavad olla liidetavaks a_i $(i = 1,2,3,4)$?

Vastused

1. Algoritmi ajalise keerukuse mõistest

1.1. Sisendparameetr $n = 16$, korrutamistehteid 15. **1.2.** Kehtib: $f(m^2) = m^2 - 1 = (m+1)(m-1)$. Sest korrutamistehete arvuks on ajalise keerukuse ($f(n) = n-1$) väärtus kohal m^2.

2. Väga kiiresti kasvava ajalise keerukusega algoritmidest

2.1. 16. Sellisel malelaual ei saa kunagi üks ratsu teise tules olla. Neljakohalisi bitijärjendeid ehk kahendarve aga on 16: $0000, 00001, 0010, ..., 1111$. **2.2.** 1122. $1122 = b - a$, kus $b = 1192$ (vähemalt 30 ratsu paigutusi) ja $a = 70$ (vähemalt 31 ratsu paigutusi). **2.3.** $f1(4,6) = 8008$; $f1(4,5) = 4368$; $f1(4,4) = 1820$; $f1(4,3) = 560$; $f1(4,2) = 120$; $f1(4,1) = 16$; $f1(4,0) = 1$; $f1(5,2) = 300$; $f1(5,1) = 25$; $f1(5,0) = 1$. **2.4.** $2, 3, 4, 5$. Suurusega 6×6 malelaual saab rahumeelselt paikneda kõige rohkem 18 ratsut ja 3×3 malelaual saab rahumeelselt paikneda kõige rohkem 5 ratsut. Seega liidetavatena tulevad kõne alla arvud $0, 1, ..., 5$. Kuid neist 0 ega 1 ei kuulu ühtegi liidetavate nelikusse, mille summa on 17 või 18. **2.5.** $9, 10, 11, 12, 13$. Suurusega 10×10 malelaual saab rahumeelselt paikneda kõige rohkem 50 ratsut ja 5×5 malelaual saab rahumeelselt paikneda kõige rohkem 13 ratsut. Seega liidetavatena tulevad kõne alla arvud $0, 1, ..., 13$. Kuid neist arvud 0, ..., 8 ei kuulu ühtegi liidetavate nelikusse, mille summa on 48 või 49 või 50.

Viited

[1] J. Kiho. *Algoritmid ja andmestruktuurid.* Kolmas, parandatud ja täiendatud trükk. TÜ, 2003, 147 lk. http://dspace.ut.ee/bitstream/handle/10062/16872/9985567676.pdf?sequence=1 (16.04.2017)