Diskreetne matemaatika ja graafid – ülesanded ja näidislahendused

1. Hulgateoreetilised tehted - võrdsuse tõestamine

a) $(P \setminus Q) \setminus R = P \setminus (Q \cup R)$
b) $(P \setminus Q) \setminus R = (P \setminus R) \setminus Q$
c) $(P \setminus Q) \setminus R = (P \setminus R) \setminus (Q \setminus R)$

1) Teisendus mingile normaalkujule (nt. ühisosade ühend, "(3)" - vt. kasutatud reeglid):
vasakud pooled – $(P \setminus Q) \setminus R = (1) = (P \cap \overline{Q}) \cap \overline{R} = (2,3) = P \cap \overline{Q} \cap \overline{R}$
a) parem pool – $P \setminus (Q \cup R) = (1) = P \cap \overline{(Q \cup R)} = (4) = P \cap (\overline{Q} \cap \overline{R}) = (2,3) = P \cap \overline{Q} \cap \overline{R}$
–> vasak pool ≡ parem pool
b) parem pool – $(P \setminus R) \setminus Q = (1) = (P \cap \overline{R}) \cap \overline{Q} = (2,3) = P \cap \overline{R} \cap \overline{Q} = (2) = P \cap \overline{Q} \cap \overline{R}$
–> vasak pool ≡ parem pool
c) parem pool – $(P \setminus R) \setminus (Q \setminus R) = (1) = (P \cap \overline{R}) \cap \overline{(Q \cap \neg R)} = (4) =$
$(P \cap \overline{R}) \cap (\overline{Q} \cup \overline{\neg R}) = (5) = (P \cap \overline{R}) \cap (\overline{Q} \cup R) = (6) =$
$((P \cap \overline{R}) \cap \overline{Q}) \cup ((P \cap \overline{R}) \cap R) = (2,3) = (P \cap \overline{R} \cap \overline{Q}) \cup (P \cap \underline{\overline{R} \cap R}) = (7) =$
$(P \cap \overline{R} \cap \overline{Q}) \cup (\underline{P \cap \varnothing}) = (8) = (P \cap \overline{R} \cap \overline{Q}) \cup \varnothing = (8) = P \cap \overline{R} \cap \overline{Q} = (2) = P \cap \overline{Q} \cap \overline{R}$
–> vasak pool ≡ parem pool

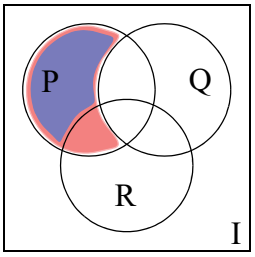
Kasutatud reeglid:
(1) $A \setminus B = A \cap \overline{B}$ – ehk $x \in A$ ja $x \notin B$; (2) kommutatiivsus; (3) assotsiatiivsus;
(4) De Morgan; (5) topelttäiend; (6) distributiivsus; (7) välistatud kolmas;
(8) $A \cap \varnothing = \varnothing$ $A \cup \varnothing = A$ $A \cap I = A$ $A \cup I = I$; $\neg A$ & $\overline{A}$ – inversioon

2) Tõestus "tõeväärtustabelite" abil (näidatud ainult *c*), 0 - $x \notin hulk$, 1 - $x \in hulk$:

P	Q	R	P\Q	(P\Q)\R	P\R	Q\R	(P\R)\(Q\R)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	1	0
0	1	1	0	0	0	0	0
1	0	0	1	1	1	0	1
1	0	1	1	0	0	0	0
1	1	0	0	0	1	1	0
1	1	1	0	0	0	0	0

3) Venn'i diagrammide abil (näidatud ainult *b*) – saadavad kujundid peavad olema "identsed".

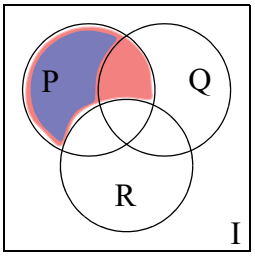
(P \ Q) \ R



P \ Q

(P \ Q) \ R

(P \ R) \ Q



P \ R

(P \ R) \ Q

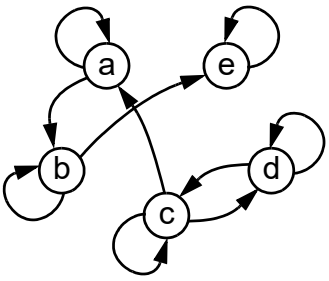
Aga kas see on ikka korrektne tõestamine?!

2. Binaarsuhte omadused

Antud on binaarsuhe R⊆AxA, A={a,b,c,d,e}, R={<a,a>, <a,b>, <b,b>, <b,e>, <c,a>, <c,c>, <c,d>, <d,c>, <d,d>, <e,e>}. Tuleb leida, kas suhe on (anti/mitte)refleksiivne, (anti/mitte)sümmeetriline ja/või (anti/mitte)transitiivne. Meeldetuletuseks:

1. Suhe on refleksiivne, kui (∀a∈A [<a,a>∈R]).
2. Suhe on antirefleksiivne, kui (∀a∈A [<a,a>∉R]).
3. Suhe, mis pole ei refleksiine ega antirefleksiivne, on mitterefleksiivne.
4. Suhe on sümmeetriline, kui (∀a,b∈A [<a,b>∈R → <b,a>∈R]), kus a ≠ b.
5. Suhe on antisümmeetriline, kui (∀a,b∈A [<a,b>∈R → <b,a>∉R]), kus a ≠ b.
6. Suhe, mis pole ei sümmeetriline ega antisümmeetriline, on mittesümmeetriline.
7. Suhe on transitiivne, kui (∀a,b,c∈A [(<a,b>∈R & <b,c>∈R) → <a,c>∈R]), kus a≠b, b≠c, a≠c.
8. Suhe on antitransitiivne, kui (∀a,b,c∈A [(<a,b>∈R & <b,c>∈R) → <a,c>∉R]), kus a≠b, b≠c, a≠c.
9. Suhe, mis pole ei transitiivne ega antitransitiivne, on mittetransitiivne.

Neid suhte omadusi saab uurida nii graafi kui ka naabrusmaatriksi abil.



	a	b	c	d	e
a	1	***1***	0	0	***0***
b	0	1	0	0	***1***
c	1	0	1	1	0
d	0	0	1	1	0
e	0	0	0	0	1

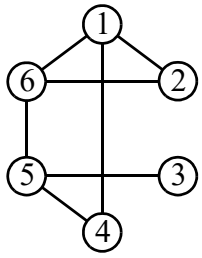
Refleksiivsusega on kõige lihtsam (vt. ka reegleid 1-3) – igal sõlmest peab minema kaar iseendale ehk naabrusmaatriksi peadiagonaalil on ainult 1-d. Antirefleksiivsuse korral ei tohiks olla mitte ühtegi kaart sõlmest iseendale ehk peadiagonaalil on ainult 0-d. Antud graaf on refleksiivne.

Sümmetria tingimuse täitmiseks peab iga kahe erineva sõlme vahel oleva kaare jaoks leiduma ka vastas-suunaline kaar. Antud juhul on see tingimus täidetud kaarte <c,d> ja <d,c> jaoks, kuid täitmata nt. kaare <a,b> jaoks. Naabrusmaatriksil tähendaks see seda, et iga peadiagonaalil mitte asuva 1 jaoks peab teisel pool peadiagonaali leiduma samuti 1 ( ). Antisümmeetria tingimuse

täitmiseks ei tohiks ühtegi sellist paari leiduda (kuid <c,d>/<d,c> on olemas). Maatriksi tähendaks see seda, et iga 1 jaoks peaks teisel pool peadiagonaali olema kindlasti 0 (nt. <a,b> e.). Kuna pole täidetud ei sümmeetria ega antisümmeetria tingimus, on antud graaf mittesümmeetriline.

Transitiivsuse tingimuse täitmiseks peab iga sõlme-kolmiku korral, kus leidub kaar esimesest sõlmest teise ja teisest kolmandasse, leiduma ka kaar esimesest sõlmest kolmandasse. Antud juhul ei leidu mitte ühtegi sellist kaare-kolmikut – on olemas <a,b> ja <b,e>, kuid puudub <a,e>; on olemas <c,a> ja <a,b>, kuid puudub <c,b>; on olemas <d,c> ja <c,a>, kuid puudub <d,a>. Rohkem vastavaid kaarte paare pole ja antud graaf on antitransitiivne. Naabrusmaatriksil on esimene mitte-leiduv kaarekolmik kaldkirjas.

3. Graafi omadused, värvimine

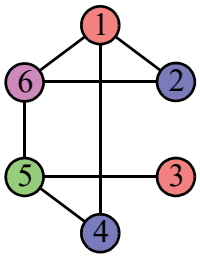


On antud joonisel olev graaf. Leida järgmised omadused: ω - klikiarv, χ - kromaatiline arv, kas graaf on planaarne, perfektne? Klikiarvu ja kromaatilise arvu määramiseks tuleks leida suurim täielik alamgraaf ja vähim arv värve selliset, et servadega ühendatud sõlmed oleks eri värvi. Planaarsus on koheselt vastatav – jah (kaare 1-4 saab viia sõlmedest 5 ja 6 vasakult mööda), perfektsuse määramiseks tuleks graafi analüüsida. Selleks on vaja teada kas kliki- (ω) ja kromaatilist-arvu (χ) või klikikatte- (κ) ja stabiilsusarvu (α). Piisab ühe paari leidmisest, sest graaf on perfektne kui ω=χ või κ=α.

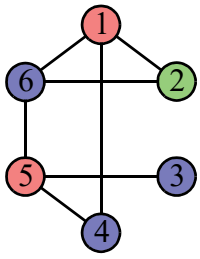
Klikiarvu (ω) leidmine – suurima täieliku alamgraafi (kliki) leidmine ja selle suuruse määramine. Täielik alamgraaf – kõik sõlmed on omavahel kaartega ühendatud. Sõlme aste (e. sisenevate/väljuvate kaarte arv) on täielikus alamgraafis ühe võrra väiksem vastava alamgraafi suurusest. Seega tuleks alustada otsimist sõlme(de)st, mille aste on suurim. Sõlmede 1, 5 ja 6 aste on 3 – suurim klikk on 4-sõlmene või väiksem. Sõlm 5 on seotud sõlmega 3, mille aste on ainult 1, seega saaks sõlm 5 kuuluda ainult kolmnurka (klikki suurusega 3), kuid sõlme 5 ülejäänud naabrid pole omavehl ühendatud. Sõlmed 1 ja 6 moodustavad koos sõlmega 2 kolmnurga. Klikiarv on seega 3. Edasi pole vaja otsida, sest suurim võimalik klikk on leitud ja arvuks piisab ainult ühest suurimast.

Kromaatilise arvu (χ) leidmine – graafi värvimine. Kolm erinevat lahenuskäiku – 2 heuristilist (ahned algoritmid) näitamaks tulemuse ebatäpsust, 1 täpne (harude ja tõkete meetod). Neljanda meetodina võiks kasutada graafi täiendi tükeldamist täielikeks alamagraafideks.

1) Ahne meetod, sõlmed valitakse nende esitamise järjekorras (1-st 6-ni). Lihtsaim meetod ja samas kõige ebatäpsem. Töö käik on lühidalt järgmine – valitakse sõlm ja kontrollitakse, kas saab värvida mõne olemasoleva värviga (kaar valitud sõlme ja mõne värvitud sõlme vahel tähendab konflikti ja võimatust kasutada vastavat olemasolevat värvi), vastasel korral luuakse uus värv. Sõlm 1 saab esimese värvi [ ]. Sõlm 2 on konfliktis sõlmega 1 (eksisteerib kaar 1-2), seega on vaja uut värvi [ ]. Sõlm 3 pole konfliktis ühegi juba värvitud sõlmega – oleks võimalik kasutada mõlemat värvi, valitakse esimene [ ]. Sõlm 4 on konfliktis sõlmega 1 – saab kasutada teist värvi [ ]. Sõlm 5 on konfliktis nii sõlmega 3 kui ka sõlmega 4, seega tuleb luua kolmas värv [ ]. Sõlm 6 on konfliktis sõlmedega 1 ( ), 2 ( ) kui ka 5 ( ). Vajalik on seega ka neljas värv [ ]. χ oleks seega 4 (ja $\omega<\chi$ ning graaf ei oleks perfektne).

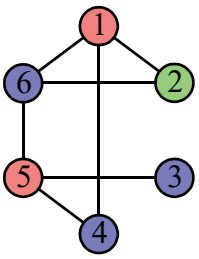


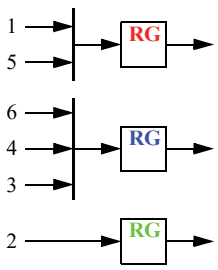
2) Ahne meetod, sõlmede valitakse nende aste järjekorras (suurima astega esimesena). Heuristika põhjendus – värvimist on parem alustada täielikest alamgraafidest ja sõlme aste viitab klikkide suurusele. Esimesena valitakse sõlm 1 - [ ]. Teisena valitakse sõlm 5 - konflikt juba värvituga puudub ja seega saab kasutada sama värvi [ ]. Kolmandana värvitakse sõlm 6, mis on konfliktis mõlema värvitud sõlmega - vajalik uus värv [ ]. Neljandana värvitakse sõlm 2, mis on konfliktis sõlmedega 1 ja 6 – vajalik kolmas värv [ ]. Eelviimasena värvitakse sõlm 4, mis on konfliktis sõlmedega 1 ja 5, kuid ei ole konfliktis sõlmega 6 – saab kasutada teist värvi [ ]. Viimasena värvitav sõlm 3 on konfliktis ainult sõlmega 5 ja kasutada saaks nii teist kui ka kolmandat värvi (valitakse esimene [ ]). Vaja on seega kokku 3 värvi (ning kuna $\omega=\chi$, siis graaf oleks perfektne).

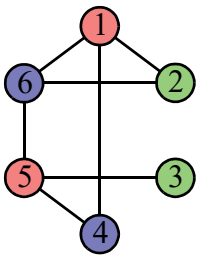


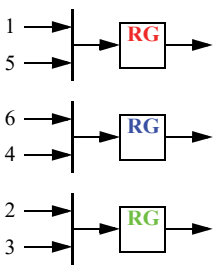
Vastus ülesandele: ω - 3, χ - 3, planaarne - jah, perfektne - jah.

Kas oleks mingit vahet, kui viimase sõlme värvimiseks valida mitte teine vaid kolmas värv? Vastus sõltuks sellest, milleks see värvimine on kasutusel. Näiteks muutujate sidumisel registritega vastaks ühte värvi sõlmede arv multipleksori sisendite arvule, st. mitmest allikast antud registrisse kirjutatakse.



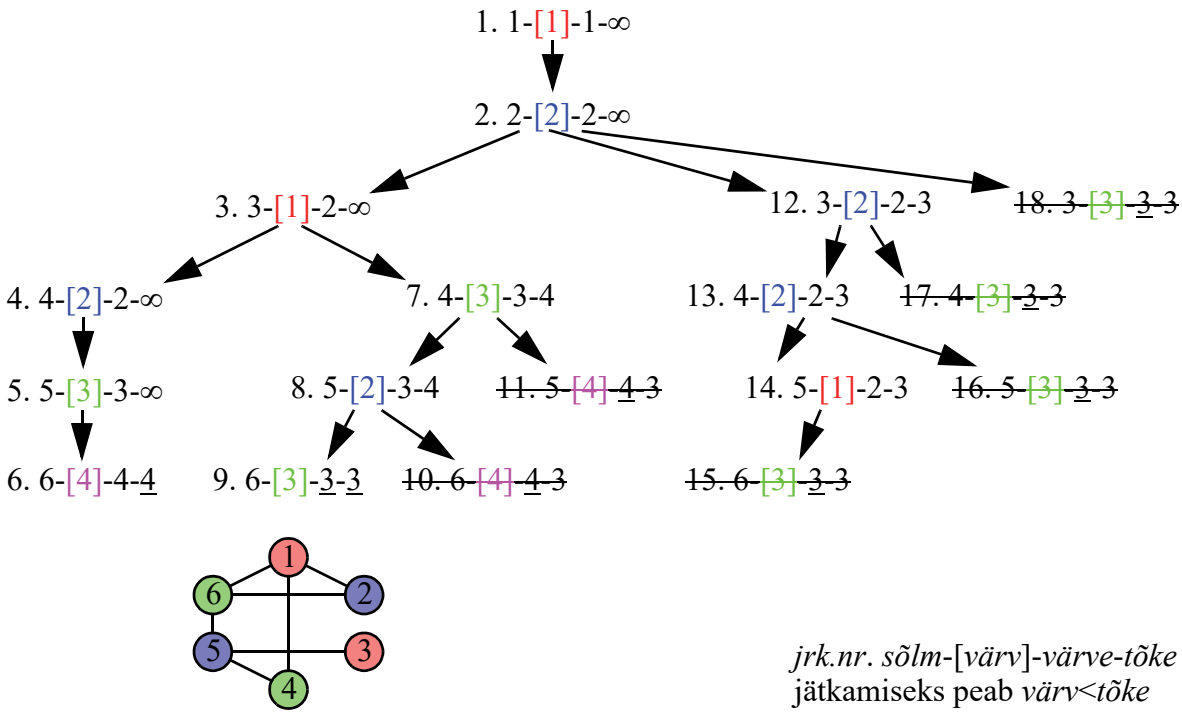






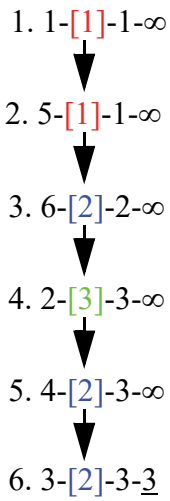
3) Harude ja tõkete meetod – täpne meetod, efektiivsus sõltub esialgse tõkke valimise meetodist. Tõkke uuendamine saab toimuda ainult siis, kui on leitud olemasolevast parem lõpp-lahend (leht hargnemiste puus). Peamine kiirusvõit kõikide variantide läbiproovimisega võrreldes seisneb selles, et kui on võimalik öelda, et osaline lahend on juba sama hinnaga kui tõkke, siis pole enam mõtet edasi otsida, sest lõpp-lahend ei saa olla odavam (värve tuleb ainult juurde). Meetodi olemus on lühidalt järgmine – iga uue (st. värvimata) sõlme valikul (järjekord pole oluline, kuid õige järjekord kiirendab lahendi leidmist) proovitakse kõiki legaalseid värvimisi (k.a. uue värvi loomine); iga legaalse osa-lahendi puhul rakendatakse sama meetodit rekursiivselt kuni kõik sõlmed on värvitud. Otsimist võib teostada nii sügavuti (valikut kontrollitakse lõpplahendus(t)eni enne, kui jätkatakse teiste valikutega) kui ka laiuti (iga valikule vastava osalahenduse hinda võrreldakse tõkkega enne, kui asutakse valikute lõpplahendusi kontrollima).
3.a) esialgne tõke - ∞, sügavuti otsimine. Esimene valik – sõlm 1 saab esimese värvi [1]. Teine valik – sõlm 2 saab kas värvi [1] või uue [2]. Legaalne on ainult [2]. Kolmas valik – sõlme 3 võib värvida kolme värviga [1], [2] ja [3] (pole konflikte sõlmedega 1 ja 2), valitakse esialgu [1], teised jätakse meelde. Neljas valik – sõlm 4 – legaalsed värvid on [2] ja [3]. Valitakse esialgu [2], [3] jääb meelde. Viies valik – sõlm 5 – legaalne on uus värv [3]. Viimane valik antud harus - sõlm 6 – legaalne on uus värv [4]. Kuna tegu on lõpplahendiga, siis saab paikka panna uue tõkke. Kogu valikute ahela võiks kirja panna järgnevalt "1/1/∞-2/2/∞-1/2/∞-2/2/∞-3/3/∞-4/4/4", kus üksik valik oleks "valitud-värv/jooksvalt-värve-kokku/tõke". Saadud lahendus langeb kokku ka antud graafi esimese värvimisega (1. ahne meetod). Edasi jätkatakse meelde jäetud osa-lahenditega. Viimane neist oli sõlme 4 värvimine [3]-ga. Sõlm 5 – legaalsed värvid on [2] ja [4]. Sõlm 6 – legaalsed värvid on [3] ja [4]. Esimene lahend annab uueks tõkkeks 3, teisel on (osa) lahendi väärtus tõkkest juba suurem. 11. lahendus-sammul on tõke väiksem osa-lahendi hinnast ja ei jätkata. 12. samm lubab sõlme 4 värvideks valida [2] ja [3]. 13. samm – sõlm 5, värvid [1] ja [3]. 14. samm – sõlm 6, värv [3]. Tõke on võrdne värvide koguarvuga ja jätkata pole mõtet, kuigi antud juhul (15.) oleks tegu veel ühe globaalse optimumiga. Ka kõikide järgnevate osalahendite puhul (16.-...) pole jätkamisel enam mõtet, sest saadav värvide arv ei tule väiksem tõkkest (ehk juba teada olevast osalahendist). Lõpptulemuseks jääb seega χ=3.

3.b) esialgne tõke leitakse mingit ahnet värvimismeetodit kasutades (nt. "2)" annaks tõkkeks 3), sõlmede valiku järjekord on määratud nende astmega (st. 1, 5, 6, 2, 4, 3) ja laiuti otsimine (esialgne tõke peab olema piisavalt hea!). Valikud 1-6 on sooritatud tõkke leidmiseks (vt. ahne meetod "2)"). 7. valik – sõlm 1 saab esimese värvi [1]. Valikud 8 ja 9 - sõlm 5, legaalsed on värvid [1] ja [2], mõlemal juhul on jätkamine võimalik. Valikud 10 ja 11 – sõlm 6, mis on konfliktis mõlema juba värvitud sõlmega, legaalne on uus värv ([2] või [3]). Neist teisel juhul on värvide arv võrdne tõkkega ja jätkamine ei oma mõtet. Valikud 12 - sõlm 2, legaalne on uus värv [3]. Ja siin algoritm lõpetab, sest mingit paremat lahendust ette pole näha. Lisamärkusena võiks mainida seda, et töö lõppes siis, kui kõik sõlmed suurimas klikis osutusid värvituks. Kogusammude arv on ligi kolmandiku väiksem, kui eelmisel variandil. Konkreetse variandi eelistamine sõltub paljudest detailidest ja eelkõige lahendatavate graafide iseloomust.

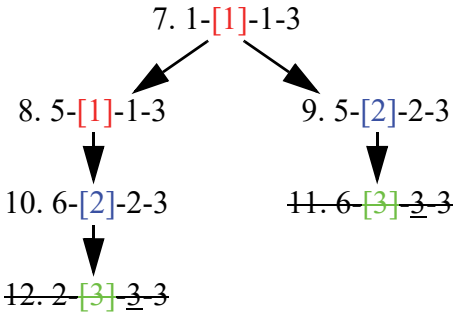


jrk.nr. sõlm-[*värv*]-*värve-tõke*
jätkamiseks peab *värv*<*tõke*

Ahne meetod tõkke algväärtustamiseks

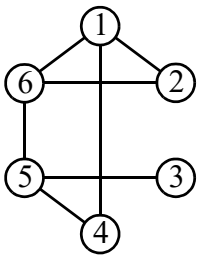


Harude ja tõkete meetod täpse lahendi leidmiseks

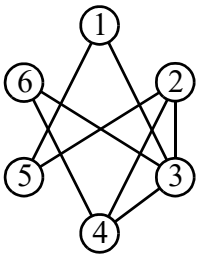


jrk.nr. sõlm-[*värv*]-*värve-tõke*
jätkamiseks peab *värv*<*tõke*

4) Sama edukalt võiks värvimise asemel hoopis tükeldada antud graafi täiendit, sest konfliktigraafi värvimine on ekvivalentne sobivusgraafi tükeldamisega. Konflikt – kahte muutujat ei saa samasse registrisse salvestada, kui neid kasutatakse samal ajal. Sobivus – kahte muutujat saab samasse registrisse salvestada, kui neid kasutatakse erinevatel aegadel.



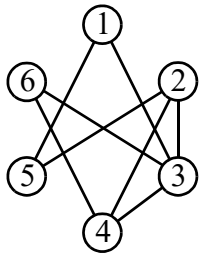
graaf



graafi täiend

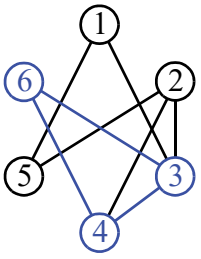
Lahendamiseks võib kasutada “vana ja head” katte leidmise meetodit, mis on kasutusel ka Quine-McCluskey meetodi teisel sammul – implikantide katte leidmisel. Selleks tuleks kõigepealt identifitseerida kõik täielikud alamgraafid ehk klikid. Kolmeseid klikke on kaks – 2-3-4 ja 3-4-6. Kaheseid klikke, mis pole kaetud kolmeste poolt, on kolm – 1-3, 1-5 ja 2-5. Klikk 3-4-6 osutub oluliseks, sest on ainus, mis katab sõlme 6. Eemaldades kaetud sõlmed (3, 4 ja 6), tekib väiksem graaf (ja tabel), mis tuleb omakorda tükeldada. Koheselt võib eemaldada klikid 2-3-4 ja 1-3, sest sõlme 2 katab klikk 2-5 ja sõlme 1 katab klikk 1-5. Kattes sõlmed 1 ja 5 klikiga 1-5 jääb ainsana katmata sõlm 2 (mis tuleks katta ühesõlmelise klikiga 2). Saadud tulemus on ekvivalentne konfliktigraafi värvimise tulemusega.

graafi täiend ja klikitabel

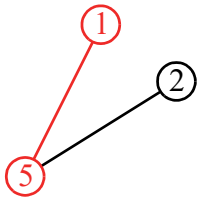


klikk	1	2	3	4	5	6
2-3-4		x	x	x		
3-4-6			x	x		x
1-3	x		x			
1-5	x				x	
2-5		x			x	

oluline klikk

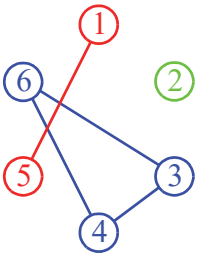


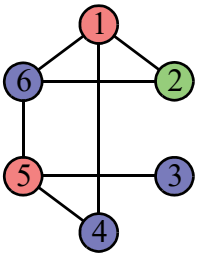
klikk	1	2	3	4	5	6
2-3-4		x	x	x		
3-4-6			x	x		x
1-3	x		x			
1-5	x				x	
2-5		x			x	



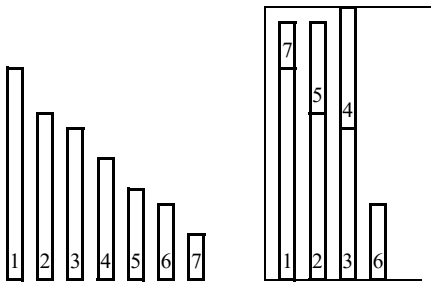
klikk	1	2	5
2-3-4		x	
1-3	x		
1-5	x		x
2-5		x	x

tulemus



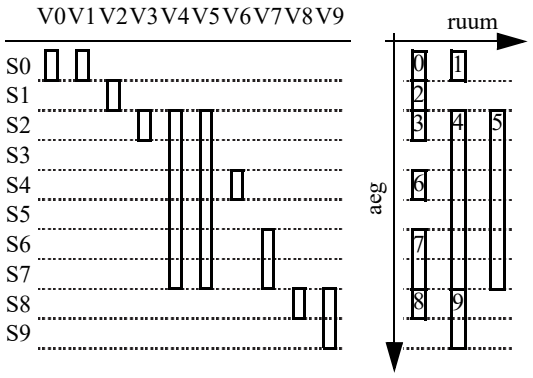


4. Pakkimine



Pakkimisülesanne seisneb objektide hulga parimas mahutamises ette antud ruumis. Sõltuvalt objektide ja ruumi olemusest eristatakse eridimensionaalseid variante. Kõige lihtsam on ühe-mõõtmeline pakkimine, kus objektidel on ainult pikkus, ruumil pikkuse piirang ja minimeeritakse “kihtide” arvu. Näitena võib tuua eri pikkusega pliiatsite karpi pakkimise, kusjuures karbi pikkus on piiratud kuid laius mitte. Pakkimisalgoritm on antud juhul väga lihtne ja annab teatud juhtudel ka täpse lahendi. Esimese sammuna objektid (pliiatsid) sorteeritakse pikkuse järgi. Seejärel võetakse esimene (st. kõige pikem) objekt ja paigutatakse ruumi äärde. Seejärel võetakse järgmine ja pannakse kas esimese objekti peale või, kui kõrgusest ei jätku, selle kõrvale. Sama järgmiste objektidega – pannakse esimesele vabale kohale, kuhu mahub.

Erijuhuks, kus lihtne algoritm annab täpse lahendi, on ülesanded, kus objektide paigutamisel on ette antud ka nende "kaugus algusest". Näiteks muutujate sidumisel registritega on muutja eluea algus ja lõpp teada. Muutuja seotakse registriga, mis on antud ajamomentidel vaba. Sisuliselt võib sellist lahendukäiku võrrelda sellega, et objektid lihtsalt lükatakse vasakule kokku. Algoritm ongi tuntud kui vasaku-serva-algoritm (left-edge algortihm). Lisaks muutujate sidumisele kasutatakse antud lähenemist ka trükkplaatide projekteerimisel radade pakketiheduse tõstmiseks.



Kui objektidel on rohkem vabadusastmeid, siis lihtne lähenemine (alla-vasakule e. bottom-left) ei pruugi täpset lahendit anda. Allpool on toodud näide, mida võib interpreteerida kastide lattu paigutamisena. Vasakpoolne tulemus on saadud alla-vasakule lähenemisega, parempoolne parendatud variant selle korrigeerimise abil. Seejuures pole mingit garantiid, et parempoolne lahend on parim (e. globaalne optimum).