TARTU ÜLIKOOL

MATEMAATIKA-INFORMAATIKATEADUSKOND

Arvutiteaduse instituut

Informaatika eriala

Sander Tiganik

Kaksiksidusate graafide visualiseerija

Bakalaureusetöö (9 EAP)

Juhendaja: Ahti Peder, PhD

TARTU 2015

Kaksiksidusate graafide visualiseerija

Lühikokkuvõte:
Töös vaadeldakse tasandiliste graafide kujutamisega seonduvaid probleeme. Uuritakse, kuidas tuvastada mõnede graafitüüpide tasandilisust. Eraldi vaadeldakse kaksiksidusaid graafe. Selle graafide klassi jaoks on teada head lahendialgoritmid. Antud töös vaadeldakse J. Hopcrofti ja R. Tarjani loodud radade lisamise meetodit. Seda kasutades lahendatakse probleem, kuidas praktiliselt luua mittetasandilise graafi võimalikult tasandilisele lähedane kujutis. Töö tulemuseks on rakendusprogramm, mis võimaldab leida sellist graafi kujutist PDF-vormingus. Sealjuures saab programmi kasutaja interaktiivselt tulemust parandada.

Võtmesõnad: tasandiline graaf, kaksiksidus graaf, radade lisamise meetod, graafi visualiseerimine, LaTeX

Visualizer of bi-connected graphs

Abstract:
The goal of this thesis is to look at the problems connected with the visualization of planar graphs, more precisely, how to determine the planarity of certain kinds of graphs. Bi-connected graphs are viewed separately. There are many well known solving algorithms for this graph class. This thesis concentrates on the path addition method created by J. Hopcroft and R. Tarjan. The algorithm is used to solve the problem of how to create the best possible planar projection of a non-planar graph. As a result of this thesis a program was created, which lets you find such a projection in PDF-format. Also the user can interactively improve the outcome.

Keywords: planar graph, bi-connected graph, path addition method, graph visualization, LaTeX

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu iseseisva töö tulemus, on esitatud Tartu Ülikooli Arvutiteaduse instituudi lõpudiplomi taotlemiseks informaatika erialal. Lõputöö alusel ei ole varem eriala lõpudiplomit taotletud.

Autor /Nimi/ .....

(allkiri ja kuupäev)

Sisukord

1 Sissejuhatus

1.1 Teema tutvustus

Graafe on võimalik kasutada mitmetel erinevatel viisidel. Graafidega on võimalik üles märgendada andmeid, objekte, objektidevahelisi suhteid, olekuid ja palju muud. Selles mõttes on graafid äärmiselt kasulikud ja universaalsed ning lubavad vajadusel näitlikustada palju erinevaid tüüpe informatsioone.
Selles töös antakse ülevaade graafide visualiseerimisest, seejuures vaadeldakse täpsemalt tasandiliste graafide kujutamisega seonduvaid probleeme. Seejärel luuakse programm, mis võimaldab graafi visuaalsest kujutisest saada LaTeX-kujul graafi kirjeldus trükkimiseks. Lõppeesmärgina nähakse programmi rakendamist struktuuride kirjelduse leidmise metoodikale [AP11], kuid võimalikke kasutusvaldkondi võib olla rohkem.
Nimetatud metoodika võimaldab teatud juhtudel leida struktuuride kirjeldusi, kasutades teatavat lähendusmetoodikat. [AP11] Selles on olulisel kohal kirjeldatava struktuuri kirjeldusega mittesobivate mudelite visuaalne ettekujutmine, milleks parim viis on graafide kujul.
Probleemiks on see, et kahe graafi võrdlemine nende omaduste suhtes, on äärmiselt raske kui graafid on teksti kujul failides. Sellest järeldub vajadus rakenduse järgi, mis suudaks võtta graafid mitmest failist ning lubaks neid graafe esitada ekraanil mõistlikul viisil. Mõistlikul viisil siinkohal tähendab, et rakendus peaks olema võimeline looma graafist kujutist, mis oleks inimsilmale kergemini hoomatav ning mis edastaks rohkem informatsiooni graafi omaduste kohta. Autori arvates on inimsilmale kõige sobilikum viis graafi kujutamiseks tasapinnaline ehk planaarne kujutis, sest siis on näha kogu graaf korraga ilma, et servad kattuksid. See lubab näha kiiresti tippudevahelisi ühendusi.
Selles töös valiti tasandilisuse omaduse tõestamiseks J. Hopcroft'i ja R. Tarjan'i *radade lisamise meetod*, see oli ka esimene lineaarse keerukusega graafi tasandilisuse tõestuse algoritm. [JH74] See algoritm töötab siiski ainult kaksiksidusate graafide peal, kuna see tagab, et algoritmi poolt tehtav töö oleks minimaalne, sest kaksiksidusust rikkuvad servad ei muuda kunagi graafi tasandilisuse omadust. Niisiis oleks nende servade töötlemine mõttetu lisatöö. Selleks et kõiki graafe oleks võimalik visualiseerida, on olemas samade autorite poolt avaldatud algoritm, mis lubab suvalise graafi jaotada tema kaksiksidusateks alamgraafideks. See tagab, et töö käigus valmiv rakendus suudab visualiseerida kõiki graafe, isegi kui need ei ole tasapinnalised.

1.2 Töö eesmärk ja uurimisküsimused

Vaja on uurida graafide tasandilisust ja nende visualiseerimist. Seejärel on töö rakendusest lähtuvalt vaja luua programm, mis lubaks sisse laadida mitu erinevat faili, mis sisaldavad samade tippude arvuga, kuid erinevate servade arvu ja konfiguratsiooniga graafi kirjeldusi ning kuvada neid graafe ekraanil. Lisaks peab programm andma kasutajale valiku rakendada sisselaetavatel graafidel algoritmi, mis aitaks kasutajal tõestada graafi tasandilisuse omaduse kehtivust. Kuna programm peab vaid kasutajat abistama, siis võib realiseeritav algoritm anda valenegatiivseid, kuid see ei tohiks kunagi anda valepositiivseid. Muuhulgas peab rakendus omama järgmisi funktsionaalsusi:

- graafi tippude liigutamine;
- graafi külgede painutamine;
- graafi väljastamine PDF-vormingus.

Kõik nimekirjas loetletud funktsionaalsused omavad eesmärki, et kasutaja saaks parandada loodud algoritmi poolt välja pakutud kujutist. Lisaks on nende eesmärgiks ka see, et kasutaja saaks vajadusel graafide joonised trükkimiseks või levitamiseks sobivale kujule viia.
Järgnevalt toome välja töö peamised uurimisküsimused.

- Kuidas teha kindlaks graafi tasandilisust?
- Kuidas visualiseerida tasandilisi graafe nii, et ei toimuks servade lõikumist?
- Kuidas saab luua algoritmiliselt mittetasandilise graafi võimalikult tasandilisele lähedane kujutis?

1.3 Töö struktuur

Töö on jaotatud viide peatükki. Peatükk "Graafide visualiseerimine" käsitleb töö teoreetilist poolt. See seletab milliseid võtteid kasutades on võimalik saavutada peatükis 1.2 püstitatud graafi tasandilisusega seotud eesmärgid.
Peatükk "Graafide visualiseerimise rakendus" tegeleb rakenduse struktuuri ja rakenduse funktsionaalsuste realiseerimise kirjeldamisega.
Peatükk "Hinnang graafide visualiseerimise rakendusele" hindab rakenduse üldist võimekust ja funktsionaalsust. Selles peatükis hindame kas ja millisel määral täidab rakendus oma eesmärke.

Peatükk "Rakenduse edasiarendamise võimalused" kirjeldab võimalusi kuidas oleks võimalik loodud rakendust edasi arendada ja paremaks muuta.
Peatükk "Kokkuvõte" võtab kokku kõik, mis antud töö jooksul tehti ning kirjeldab töö tulemusi.

2 Graafide visualiseerimine

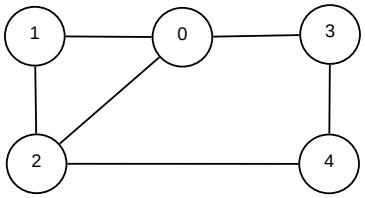
Selles peatükis selgitame vajaliku osa graafi tasandilisuse tõestamisest ja sellega seotud toimingutest ning algoritmidest. Lisaks kirjeldame kuidas on võimalik, kasutades graafi tasandilisuse tõestuse protsessis tekkinud andmestruktuure, luua ekraanile selline graaf, mis oleks abistav joonis kasutajale, kes soovib luua graafist tasapinnalist kujutist.

2.1 Kaksiksidusad graafid

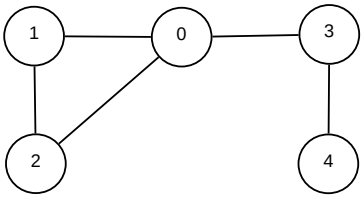
Olgu graaf G = (V,E) suunamata graaf, kus V on graafi G tippude hulk ja E tema servade hulk. Graafi nimetatakse *sidusaks*, kui iga suvalise tipupaari $u, v \in V$ korral on võimalik liikuda tipust u tippu v kasutades selleks ainult servi hulgast E. [AB03] Graafi nimetatakse *kaksiksidusaks*, kui ükskõik millise tipu $v \in V$ ja kõikide selle tipuga intsidentsete servade eemaldamise korral, on graaf endiselt sidus. [BiCa]

Lause 1. - Kaksiksidusa n-tipulise ($n \geq 3$) graafi iga tipp kuulub mingisse tsüklisse.

Tõestus - On selge, et iga tipu aste on vähemalt 2. Kui see nii ei oleks, siis leidub mingi tipp, mille aste on 1. Selle tipu naabertipp on sellisel juhul kaksiksidusust rikkuv. Seega graafi iga tipu aste on vähemalt 2. Oletame vastuväiteliselt, et mingi tipp v ei kuulu ühessegi tsüklisse. Kuna v tipu aste on vähemalt 2, siis on see tipp ilmselt kaksiksidusust rikkuv. Seega iga tipp kuulub mingisse tsüklisse.



Joonis 1: Kaksiksidus graaf [BiCb]

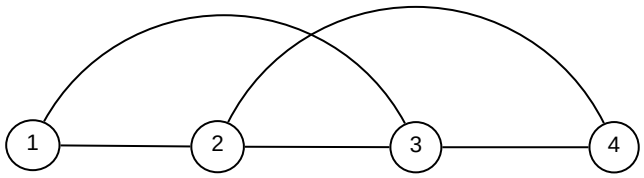


Joonis 2: Graaf, mis ei ole kaksiksidus (kaksiksidusust rikuvad tipud 0 ja 3) [Mit]

2.2 Graafi jaotamine kaksiksidusateks komponentideks

Aastal 1973 pakkusid J. Hopcroft ja R. Tarjan välja tippude arvu n suhtes Θ(n) ajalise keerukusega algoritmi mille abil on võimalik jaotada suvaline graaf tema

kaksiksidusateks komponentideks. [JH73] Kirjeldatud algoritm põhineb sügavutiläbimise meetodil. Läbides graafi sügavuti, tuleb meelde jätta antud tipu järjekorranumber sügavuti läbimisel ja selle tipu madalaim saavutatav väärtus. *Madalaim saavutatav väärtus* on sügavuti läbimise järjekorra madalaim väärtus, kuhu on võimalik jõuda antud tipust, kasutades selleks ükskõik millist serva välja arvatud serva, mille kaudu sellesse tippu jõuti (joonis 3).



Joonis 3: Kui sügavuti läbimine toimus järjekorras 1→2→3→4, siis tipu *4* madalaim väärtus on 2, tipu *3* madalaim väärtus on 1 ja tipu *2* madalaim väärtus on 2 ise

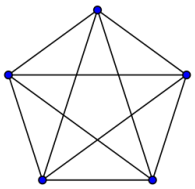
Kui sügavuti läbimine jõuab tippu, mille kõik järglased on töödeldud, hakatakse sellest tipust tagastama madalaimat väärtust tema eellastele. Kui ei leidu madalamat väärtust kui tipu enda sügavuti läbimise järjekorra number, siis tagastatakse antud tipu järjekorra number. Kui mingil hetkel saab mõni tipp tagastusväärtusena arvu, mis on suurem või võrdne kui antud tipu sügavuti läbimise järjekorra number, siis rikub see tipp kaksiksidususe omadust. Viimasena tuleb töödelda juurtippu, ehk tippu millest algoritm alustab. Juurtipp rikub kaksiksidusust ainult eeldusel, et temast saab alguse kaks või rohkem erinevat sügavuti läbimise haru. Kui juurtipust saab alguse vaid üks haru, siis on juurtipp lehttipp ning tema eemaldamine ei riku kaksiksidusust. Kui aga juurtipust saab alguse kaks või rohkem haru, siis saab juurtipu eemaldades lahutada mõlemad harud eraldiseisvateks komponentideks, mis tähendab et juurtipp rikub kaksiksidususe omadust.

2.3 Tasandilised graafid

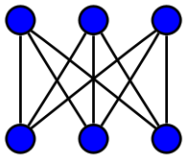
Graafi nimetatakse *tasandiliseks* ehk *planaarseks*, kui teda on võimalik joonistada tasandile nii, et tema servad ei lõiku väljaspool tippe. [AB03]

Teoreem 1.1 (Euler, 1750) - Kui G on sidus tasandiline graaf ning n,m ja f vastavalt tippude, servade ja tahkude arvud tema joonisel, siis $n + f - m = 2$. [AB03]

Kasutades Euler'i teoreemist tulenevat järeldust saab öelda, et graafid K_5 (viie tipuga täisgraaf) ja $K_{3,3}$ (6 tipuga ja 9 küljega graaf, milles ei ole paarituarvulise pikkusega tsükleid) ei ole tasandilised (joonised 4, 5). [AB03]



Joonis 4: Graaf K_5 [K5]

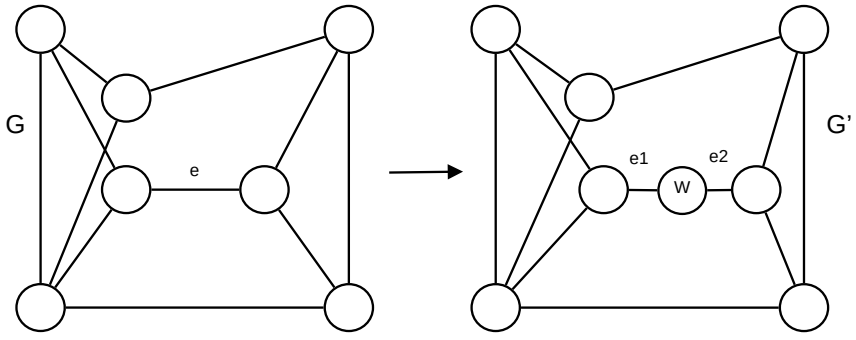


Joonis 5: Graaf $K_{3,3}$ [K33]

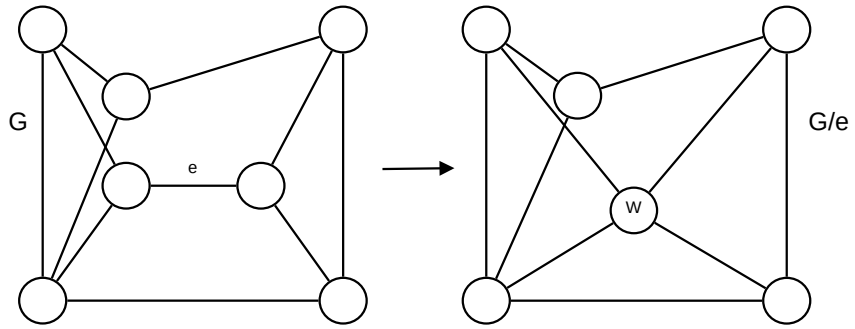
Kuna joonistamise mõiste ei ole aga matemaatiliselt range, siis toome järgnevalt kaks teoreemi, mis määravad täpselt graafi tasandilisuse tingimused. Siinkohal tuleb mainida, et selles töös olevad mõisted homöomorfism, graafi külgede poolitamine (joonis 6) ja graafi külgede kokkutõmbamine (joonis 7) langevad kokku A. Buldas, P. Laud ja J. Villemson poolt kirjutatud raamatus "Graafid" antud definitsioonidega. [AB03]
Graafe G_1 ja G_2 nimetatakse *homöomorfseteks*, kui leidub mingi selline graaf G, et nii G_1 kui ka G_2 on saadavad graafist G servade poolitamise teel. [AB03]

Teoreem 1.2 (Kuratowski) - Graaf on tasandiline parajasti siis, kui ükski tema alamgraaf pole homöomorfne graafiga K_5 või $K_{3,3}$. [AB03]

Teoreem 1.3 (Wagner) - Graaf on tasandiline parajasti siis, kui ükski tema alamgraaf pole kokkutõmmatav graafiks K_5 või $K_{3,3}$. [AB03]



Joonis 6: Serva e poolitamine [AB03]



Joonis 7: Serva e kokkutõmbamine [AB03]

2.4 Graafi tasandilisuse tõestamine radade lisamise meetodil

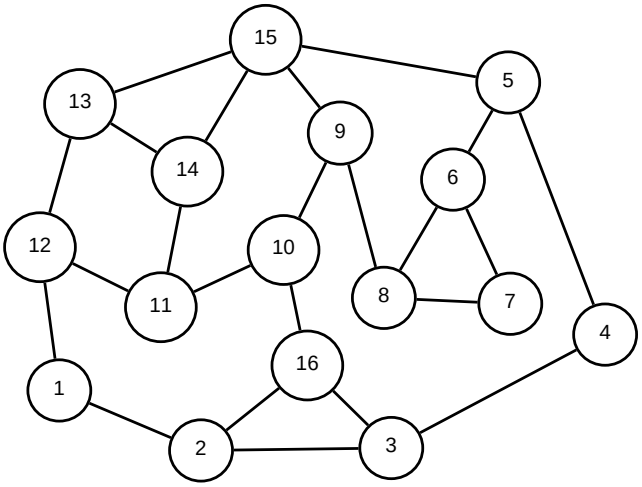
2.4.1 Radade lisamise meetod

Aastal 1974 pakkusid J. Hopcroft ja R. Tarjan välja algoritmi millega on võimalik efektiivselt tõestada suvalise kaksiksidusa graafi tasandilisust. [JH74] Selle tippude arvu n suhtes Θ(n) ajalise keerukusega algoritmi nimeks sai *radade lisamise meetod* ning see põhineb, nii nagu ka kaksiksidusateks komponentideks jaotamise algoritm, graafi sügavuti läbimisel.

Nagu ülalpool mainisime, on algoritmi eelduseks see, et graaf, mille tasandilisust hakkame tõestama, on kaksiksidus. Kuna aga suvalist graafi on võimalik jaotada kaksiksidusateks komponentideks, saab seda algoritmi rakendada igal graafil. Selleks tuleb alggraaf jaotada tema kaksiksidusateks komponentideks ning meeles pidada eemaldatud küljed. Edasi tuleb rakendada radade lisamise meetodit igal leitud kaksiksidusal komponendil. Kui tulemuseks on, et kõik kaksiksidusad komponendid on tasandilised, võib väita, et alggraaf ise on ka tasandiline, sest alggraafi kaksiksidususe omadust rikkuvate servade alggraafi tagasi lisamine ei saa rikkuda tasandilisuse omadust.

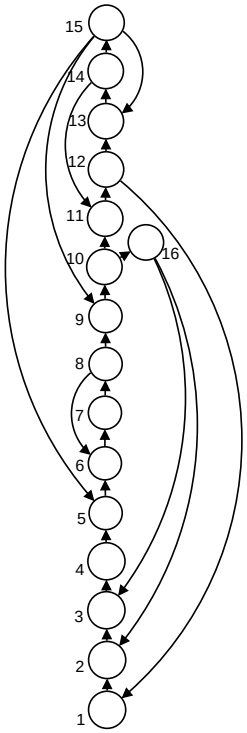
Algoritm algab sügavuti läbimise teostamisega sisendgraafil. Sügavuti läbimise teostamisel jäetakse meelde sügavuti läbimise algoritmi iga tipu korral selle läbimise järjekorra number.

Kui sügavuti läbimine on jõudnud tipuni, mille kõik järglased on juba töödeldud, hakatakse üles märkima nn. tagaservi, alustades sügavuti läbimise hetkel töötluses oleva haru viimati töödeldud tipust. *Tagaservadeks* nimetatakse graafi kõiki neid servi, mida mööda ei ole veel senini käidud, kuid mille mõlemad otstipud on töödeldud. Kui graafi sügavuti läbimise puu peaks mingis kohas hargnema, mille korral sügavuti läbimine nõuab tagasi liikumisel teise haru läbi käimist, siis käiakse enne läbi teine haru ja seejärel hakatakse taas

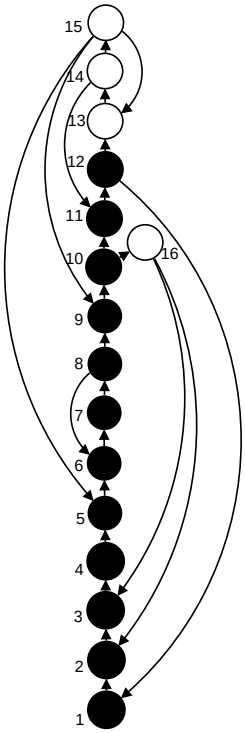


Joonis 8: Algne graaf, märgitud on sügavuti läbimise järjekorra numbrid [Kug]

üles märkima tagaservi teise haru lõpust. Kui sügavuti läbimise algoritm lõpetab, tekib andmestruktuur, mida on võimalik kujutada nii, et see meenutab ″palmipuud″, nii nagu seda nimetasid J. Hopcroft ja R. Tarjan oma artiklis (joonis 9). [JH74]



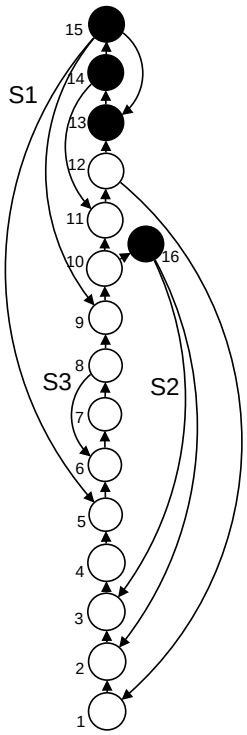
Joonis 9: ″Palmipuu″ struktuur. Tagaservad liiguvad kõrgematelt tipu väärtustelt madalamatele [Kug]



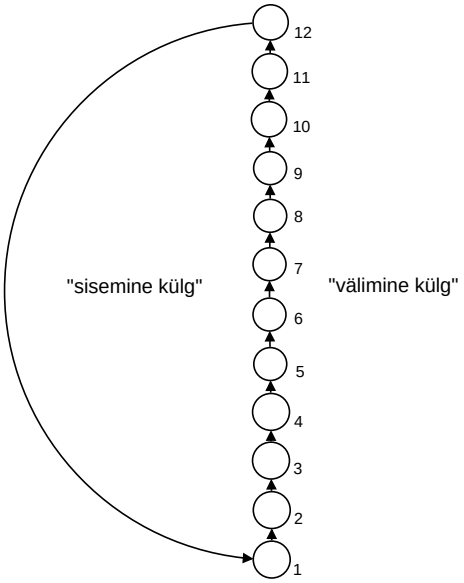
Joonis 10: Lühim tsükkel, mis alggraafis algab tipust 1 (1→2→...→11→12) [Kug]

Kui palmipuu struktuur on leitud, tuleb leida graafi lühim tsükkel (joonis 10). [JH74, Pla] Tsükli moodustavad kõik tipud alates tipust mille väärtus on 1 (kaasa arvatud) kuni esimese tipuni millel leidub tagaserv, mis liigub tagasi algustippu (kaasa arvatud). Tsükkel moodustab tippudest ja servadest ahela, mida on võimalik kujutada ringjoone peal, jaotades graafiliselt pinna kaheks eraldi alaks, mida võib nimetada ″sisemiseks″ ja ″välimiseks″ küljeks (joonis 12). [JH74]

Järgmiseks tuleb leida radade lisamise meetodile iseloomulikud rajad. Radade leidmiseks tuleb eemaldada palmipuu stuktuurist tsükkel. Pärast tsükli eemaldamist jääb alles mingi hulk segmente, mis ise koosnevadki radadest (joonis 11). Näiteks joonisel 11 on kolm eristatavat segmenti, mille nimed on $S1$, $S2$ ja $S3$. Segment $S1$ koosneb neljast erinevast rajast, milleks on $r1 = \{12, 13, 14, 15, 5\}$, $r2 = \{15, 9\}$, $r3 = \{15, 13\}$ ja $r4 = \{14, 11\}$. Segment $S2$ koosneb kahest rajast, milleks on $r5 = \{10, 16, 2\}$ ja $r6 = \{16, 3\}$ ning segment $S3$ koosneb ühes rajast milleks on $r7 = \{8, 6\}$. [JH74, Pla]



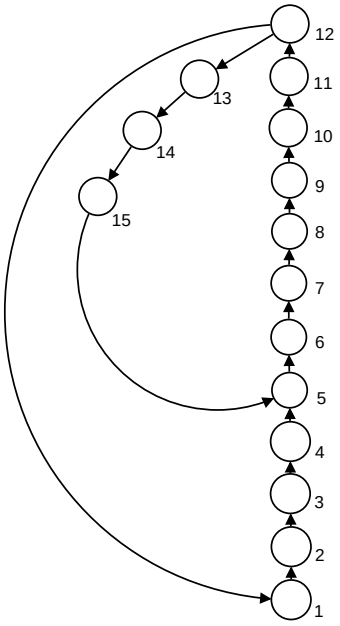
Joonis 11: Palmipuu struktuurist eemaldatud tsükkel jaotab joonise segmentideks [Kug]



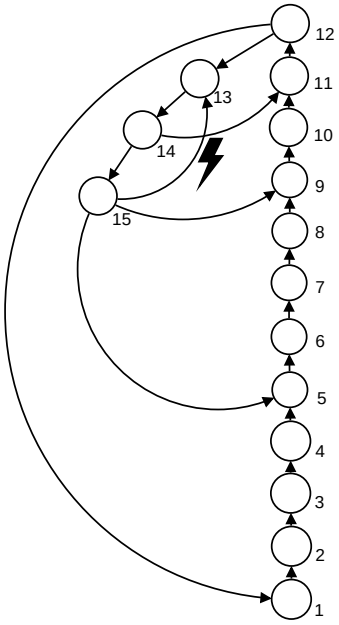
Joonis 12: Algoritmi algus [Kug]

Viimasena tuleb tsüklisse hakata sisestama radasid. Radasid sisestatakse nende sügavuti läbimise järjekorra järgi. Kui on valik kahe väärtuse vahel, valitakse alati see rada, mille lõppväärtus on madalam. Eelmises näites toodud radade

$r5$ ja $r6$ põhjal valitakse esialgu rada $r5$, sest raja $r5$ lõppväärtus 2 on madalam kui raja $r6$ lõppväärtus 3. [JH74, Pla]
Radasid lisatakse alati tsükli sisemisele küljele (joonised 12 ja 13). Tsükli sisemiseks küljeks on ala, mis jääb ülalpool kirjeldatud ringjoonel asuva ahela sisepinnale. Kui mõned rajad ühe segmendi sees lõikuvad, siis proovitakse lahendada konflikt (servade lõikumine) segmendi siseselt. Kui konflikti lahendamine ebaõnnestub, siis ei ole graaf tasandiline (joonised 14 ja 15). [JH74]

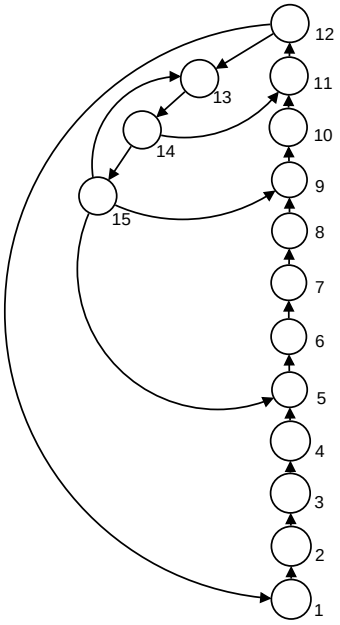


Joonis 13: Raja {12,13,14,15,5} lisamine tsüklisse [Kug]

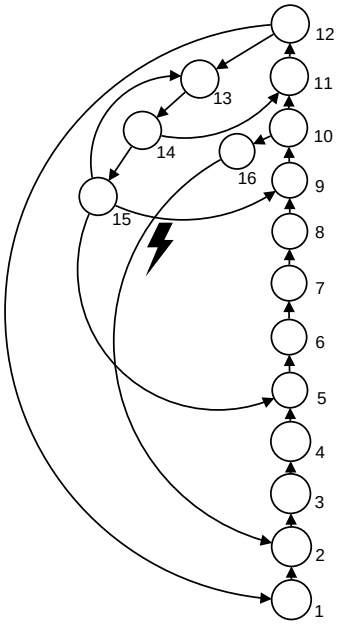


Joonis 14: Konflikt [Kug]

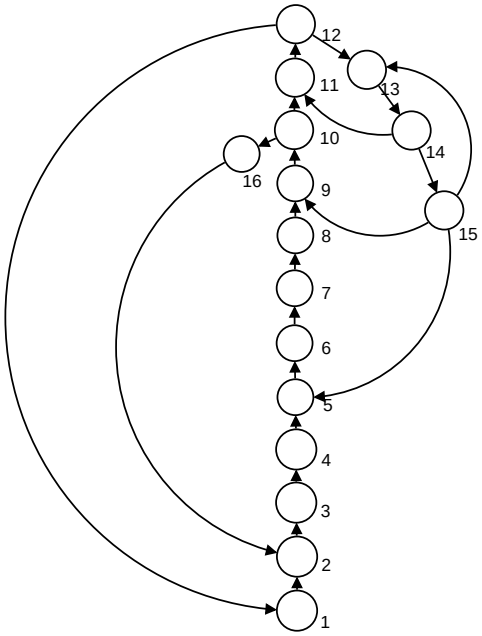
Kui toimub konflikt kahe segmendi vahel, siis viiakse olemasolevad tsüklisse sisestatud rajad tsükli teisele ″küljele″ (joonised 16 ja 17). Pärast seda lisatakse uus segment tsükli ″sisemisele″ küljele ning kontrollitakse, kas teisele poolele (välisküljele) viidud olemasolevad rajad on vastuolus mõne seal küljel oleva rajaga. Kui toimub uus konflikt, proovitakse täpselt sama lahendust uuesti erinevusega, et seekord tuuakse välimisel küljel olevad rajad sissepoole. Kui tekib endiselt uus konflikt, siis ei ole graaf tasandiline, sest ei leidu ühtegi kombinatsiooni segmentide jaotusest külgede vahel nii, et nad ei looks konflikti. [JH74]
Kui jätkata sama põhimõtet on tulemuseks üks kahest võimalusest. Algoritm kas teatab konfliktist, mida ei ole võimalik lahendada ning järeldus on, et graaf ei ole tasandiline või kõik rajad sisestatakse tsüklisse ning algoritm lõpetab edukalt, mis tähendab, et graaf on tasandiline (joonis 18).



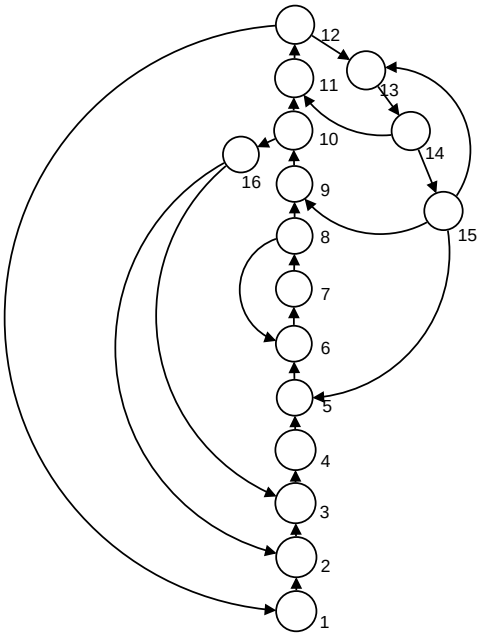
Joonis 15: Konflikti lahendus [Kug]



Joonis 16: Segmentidevaheline konflikt [Kug]



Joonis 17: Segmentidevahelise konflikti lahendus [Kug]



Joonis 18: Tulemus [Kug]

2.4.2 Radade lisamise meetodi arvuti realisatsioon

Programmi loomiseks, mis kasutab radade lisamise meetodit, tuleb kasutusele võtta mõned abistruktuurid ja lihtsustavad põhimõtted selleks, et peatükis 2.4.1 kirjeldatud algoritmi oleks lihtsam realiseerida. Järgnevalt kirjelda-

takse milliseid võtteid kasutas autor selleks, et graafide visualiseerimise programmis realiseerida radade lisamise meetodit. Autor rõhutab siinkohal, et radade lisamise algoritm on väga keeruline ning töö käigus loodud realisatsioon ei ole täielik, mis tähendab, et algoritm on võimeline tegema vigu. Vead võivad tekkida valenegatiivsete kujul, kuid mitte valepositiivsete kujul. Eesmärgiks oli luua algoritm, mis suudaks hinnata lihtsamate graafide tasandilisust ning aidata kasutajat võimalikult palju tasandilisuse omaduse tõestamisel.
Selleks, et realiseerida algoritmi kasutatakse graafide visualiseerimise rakenduses kahte klassi, mille nimedeks on *Cycle* ja *Path*, mis vastavad radade lisamise algoritmi *tsükkel* ja *rada* mõistetele. Mõlemad klassid on realisatsioonilt sarnased selle poolest, et nad sisaldavad kahte nimekirja (inglise keeles *array*), mille nimedeks on *Cycle* klassi puhul *inside* ja *outside* ning *Path* klassi puhul *left* ja *right*. Nimede valik on tingitud sellest kuidas antud struktuurid jaotavad graafilises ülesmärgenduses alasid. [Tay08]
Graafide visualiseerimise rakenduses eristatakse kolme eri liiki radasid. Nendeks liikideks on:

- tsükli rajad - näiteks joonis 18 rajad {8,6} ja {10,16,2};
- rajapealsed rajad - näiteks joonis 18 rada {15,13};
- ühendavad rajad - näiteks joonis 18 rajad {14,11} ja {16,3}.

Kõik kolm liiki on eristatavad nende algus- ja lõpptipu kaudu. Tsükli rada algab ja lõpeb lühimal tsüklil, rajapealne rada algab ja lõpeb teise raja peal ning ühendava raja omaduseks on, et see algab mingi raja peal, aga lõpeb leitud lühimal tsüklil. Meeldetuletuseks siinkohal, et Lause 1 tõttu kaksiksidusa graafi iga tipp kuulub mingisse tsüklisse.
Kõigi kolme rajaliigi sisestamisel leitud lühimasse tsüklisse tuleb silmas pidada järgnevalt kirjeldatavaid omadusi. Kui lisatakse tsükli rada, on võimalik konflikt saavutada kahel viisil. Esimeseks viisiks on, et sisestatav tsükli rada lõikab mõnda olemasolevat tsükli rada. Selle konflikti kontrollimiseks saab ära kasutada sügavuti otsingu ajal määratud tippude väärtusi. Oletame, et sisestatav tsükli rada algab tipus väärtusega x ja lõpeb tipus väärtusega y. Selleks et tuvastada konflikti, tuleb läbi vaadata tsükli rajad, mis asuvad samal tsükli küljel. Oletame et parajasti vaadeldava tsükli raja algus- ja lõpptipp on vastavalt z ja w. Kuna sügavutiotsing on järjestanud tsükli ahela väärtused suuruse järjekorras, piisab kontrollida, kas $x > z > y > w$ või $z > x > w > y$. Kui üks nendest võrratustest on tõene, siis on kaks tsükli rada omavahel konfliktis.

Teiseks viisiks kuidas sisestatav tsükli rada saab luua konflikti on kui mõni ühendava raja lõpptipp asub sisestatava tsükli raja algus- ja lõpptipu vahel. Ehk kehtib võrratus $x > w > y$, kus w on ühendava raja lõpptipp. [Pla]
Kui lisatakse raja peale rada, on võimalik, et tekib konflikt kahel erineval viisil. Esimeseks võimaluseks on, et raja peal leidub juba rada, mis on konfliktis lisatava rajaga. Sellisel juhul kehtib täpselt sama võrratus, mis kehtis ka tsükli radade puhul ehk $x > z > y > w$ ja $z > x > w > y$, kus x ja y on sisestatava raja algus- ja lõpptipp ning z ja w on läbivaadatava raja algus- ja lõpptipp. Teiseks võimaluseks kuidas raja peale raja lisamine võib tekitada konflikti on, kui leidub ühendav rada mille algustipp asub sisestatava raja algustipu ja lõpptipu vahel ehk kehtib uuesti eelmises lõigus toodud võrratus $x > w > y$, kus w on ühendava tipu algustipp. [Pla]
Kui lisatakse ühendav rada, siis on teostatav konflikti kontroll analoogiline eelnevate kontrollidega ainult, et vastupidi. Selle asemel, et kontrollida, kas leidub mõni konfliktis olev ühendav rada, tuleb vaadata, kas leidub mõni tsükli rada millega sisestatav ühendav rada on konfliktis või kas leidub mõni raja pealne rada, mis on konfliktis sisestatava rajaga.
Kasutades neid põhimõtteid, on võimalik luua radade lisamise algoritmi realisatsioon, mis suudab väga suure hulga graafide tasandilisust hinnata. Siiski nagu ülalpool mainitud, ei ole neid põhimõtteid kasutav realisatsioon täielik. Kuid see ei anna ka valepositiivseid, kui märkida kõik mitte äratuntavad olukorrad kui lahendamatud konfliktid.

2.5 Graafi kujutise loomine

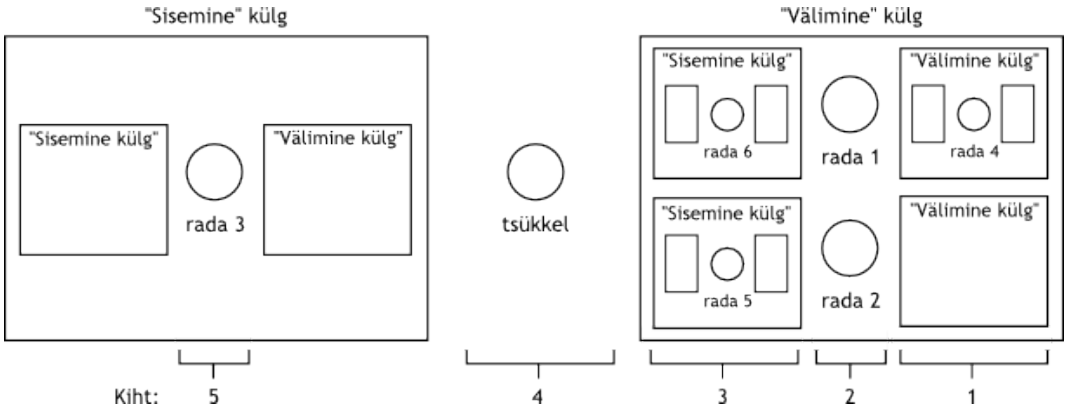
Kasutades J. Hopcrofti ja R. Tarjani *radade lisamise meetodit* on võimalik tõestada või ümber lükata suvalise kaksiksidusa graafi tasandilisust. [JH74] Siiski ei anna see algoritm kasutajale mingit informatsiooni selle kohta, kuidas on võimalik seda graafi kujutada pinnal niimoodi, et tasandilisuse omadust ei rikutaks. Selle asemel algoritm ainult kinnitab, et selline kujutis graafist on olemas eeldusel, et graaf on tasandiline või et selline kujutis puudub, kui graaf ei ole tasandiline. Küll aga on võimalik kasutada algoritmi käigus tekkivaid andmestruktuure koos andmetega selleks, et tasapinnale luua hea lähend tasandilisest graafist, mis abistab kasutajat ülesandes leida tasandiline kujutis. Järgnevalt tuuakse välja üks põhimõte algoritmi käigus tekkinud andmestruktuuride ja andmete kasutamiseks, et luua tasandilisuse lähedane graafi kujutis. Radade lisamise algoritmi tulemuseks, nii nagu see kirjeldatud peatükis 2.4, on massiiv *Cycle* objektidest, mis sisemiselt omavad puutaolist struktuuri.

Kuna kõik massiivis sisalduvad objektid esindavad omaette kaksiksidusaid alamgraafe vastavalt põhimõttele, et enne radade lisamise algoritmi kasutamist tuleb jaotada graaf tema kaksiksidusateks komponentideks, siis võib igat tsüklit massiivis käsitleda kui eraldiseisvat graafi, mis ei ole teistega seotud. See lubab lihtsustada graafi joonistamist jaotades kogu olemasoleva pinna n võrdseks osaks. Järgnev autori poolt välja pakutud valem näitab kuidas leida, mitmeks alaks tuleb pind jaotada, et mahutada ära kõik alamgraafid massiivis, kui massiivi pikkuseks on l:

$$n = \begin{cases} l, & \text{kui } l \text{ on paaris ;} \\ l+1, & \text{vastasel juhul .} \end{cases} \tag{1}$$

Valemist (1) selgub, et etteantud pind, jaotatakse n võrdseks osaks, kusjuures n on alati paarisarvuline väärtus. See on tingitud põhjusest, et ristkülikukujulise pinna jaotamine paarituarvuliseks võrdseks osaks on ebapraktiliselt keeruline. Edasi saab hakata võtma massiivist ükshaaval tsükleid, et täita nüüdseks juba n osaks jaotatud tasandi pinda alamgraafi tippudega. Igal tsüklil saab olla täpselt kaks olekut: tasandilisus tõestatud või tasandilisuse tõestamine ebaõnnestus.
Kui algoritmil ei õnnestunud tasandilisust tõestada, ei tähenda see, et alamgraafi ei saaks kuvada pinnale, aga kuna algoritmi väitel tasandilist kujutist sellest alamgraafist ei leidu, ei ole mõtet sellele alamgraafile rakendada tasapinnalise kujutise loomise põhimõtet. Niisiis võib võtta kõik selles alamgraafis sisalduvad tipud ning jaotada need ära suvaliselt sellele alamgraafile ettenähtud pinna osale.
Kui suudeti tõestada antud alamgraafi tasandilisust, saab ära kasutada alamgraafi andmete puutaolist struktuuri selleks, et luua nii-öelda "tasemeline" kujutis antud alamgraafist. Nii nagu peatükis 2.4.2 on selgitatud, jaotuvad nii tsükkel kui ka rajad "sisemiseks" ja "välimiseks" pooleks. Kasutades seda teadmist koos informatsiooniga puu struktuurist, on võimalik luua rekursiivne funktsioon, mis loendab ära "tasemete" arvu antud alamgraafis. Tasemete arvu loendamine käib põhimõtte järgi, et esialgu kutsutakse välja rekursiivne funktsioon tsüklil. Edasi kutsutakse sama funktsioon rekursiivselt välja kõikidel radadel, mis on "välimisel" küljel. Iga funktsioon tagastab arvulise väärtuse, mis näitab, mitu taset oli selles rajas ning nende seast valitakse suurima väärtusega arv. Nüüd liidetakse sellele arvule juurde 1, kuna arvesse tuleb võtta ka tipp, milles parajasti viibitakse. Edasi kutsutakse välja täpselt sama rekursiivne funktsioon

”sisemise” külje radadel ning uuesti valitakse suurima väärtusega arv. Järgnevalt see väärtus tagastatakse funktsioonile, mis ta kutsus. Nüüd kus kõik võimalikud teed tsüklist alates on läbi käidud, võib öelda, et antud alamgraafi tasemete arv ongi see väärtus, mis tagastatakse tsükli funktsioonist (joonis 19).



Joonis 19: Näide ”tasemelisest” jaotusest

Kui alamgraafide tasemete arv on leitud, on võimalik väga väikeste loendamise funktsiooni koodi muudatustega luua uus funktsioon, mis tagastaks kõik tipud, mis asuvad tasemel i antud graafis. Nüüd, kus on võimalik jaotada alamgraafi tasemete kaupa, on ka võimalik teda joonistada tasemete kaupa. Joonistamise muudab võimalikuks see, et radade lisamise algoritm suutis tõestada antud alamgraafi tasandilisust, niisiis lõpetas algoritm edukalt radade põimimise tsükli sisse. See tähendab, et kõik rajad mis asuvad tsükli sees, on juba oma õige koha peal ning kasutades tasemelist joonistamise võtet, ei saa tekkida mitte ühtegi ülemäära suurt viga. Ülemäära suureks veaks nimetab autor rada, mis rikub peatükis 2.4.2 kirjeldatud realisatsiooni reegleid.
Joonistamine toimub nagu ülalpool juba mainitud, tasemete kaupa, ringjoonte peale. Kuna alamgraafi tasemete arv k on teada, saab jaotada alamgraafile eraldatud pinna ala k üksteise sees võrdsetel kaugustel asuvateks ringjoonteks. Joonistamist tuleb alustada kõige välimisest ringjoonest, kuna rekursiivne funktsioon, mis tagastab alamgraafi tippe, tagastab argumendi 1 korral kõige välimise taseme tipud. Edasi tuleb läbi käia kõik alamgraafi (algraafi kaksiksidus komponent) tasemed ning joonistada kõik vaadeldavas tasemes sisalduvad tipud vastavale ringjoonele joonises. Selleks tuleb ringjoon jaotada l võrdseks osaks, kus l on antud hetkel vaadelava taseme tippude massiivi pikkus. Seda on lihtne saavutada kasutades selleks rotatsioonimaatriksit, keerates iga tipu temale vastavale kohale ning teadmist, et ringis on 360°.

Kui kõik tasemed antud alamgraafis on joonistatud, tuleb edasi liikuda järgmisele pinna alale ning võtta tsüklite massiivist järgmine tsükkel, ning korrata kujutise loomise algoritmi, kuni kogu tsüklite massiiv on läbitud. Viimase tegevusena tuleb lisada graafi tagasi eemaldatud servad, mis rikkusid algse graafi kaskiksidususe omadust. Nende servade tagasilisamine ei muuda graafi tasandilisust, niisiis ei tule nende tagasilisamisel kontrollida mingeid tingimusi. Pärast eemaldatud servade tagasilisamist on graafi kujutis valmis.

Siinkohal tahab autor mainida, et ülalpool kirjeldatud kujutise loomise põhimõte on väga paindlik. Isegi kui graafi tasandilisuse tõestus ebaõnnestub, joonistab algoritm ikkagi nii palju alamgraafe algsest graafist tasandiliselt, kui võimalik ja neid alamgraafe, millele tasandilisuse tõestust ei leidu, joonistatakse suvalisi väärtusi kasutades, nendele ettenähtud pinnaaladele. See tähendab, et isegi kui programm peaks tegema vea, on kasutajal tunduvalt lihtsam kontrollida lõpptulemust. Ka tähendab selline joonistuse eeskiri, et ei ole olemas mitte ühtegi graafi, mida ei saaks kuvada.

3 Graafide visualiseerimise rakendus

Selles peatükis räägitakse töö käigus valminud graafide visualiseerimise rakendusest. Sealjuures rakenduse struktuurist, funktsionaalsusest ja põhimõtetest.

3.1 Rakenduse kirjeldus

Graafide visualiseerimise rakendus (koodnimega GraphDrawer) on programm, mis võimaldab kuvada ekraanile graafe. Rakendus võimaldab graafe sisse laadida kahel viisil, kasutades selleks DIMACS-vormingus graafi kirjeldust või rakenduse enda poolt kasutatavat failiformaati, mida kirjeldatakse pikemalt peatükis 3.3. [fDMD93] Esimesel juhul antakse kasutajale võimalus enne graafide kuvamist valida, kas ta soovib rakendada graafi kirjeldusele tasapinnalisuse tõestamise algoritmi või soovib ta tippude jaotamist ekraanile suvaliselt. Laadides sisse graafe, mis on rakenduse enda failiformaadis, on kõik tippude asukohad juba kindlaks määratud, sealjuures täiendavaid küsimusi kasutajale ei esitata. Pärast graafi avamist rakenduses on kasutajal hulk erinevaid funktsionaalsusi, mida ta saab kasutada graafide manipuleerimiseks ja väljastamiseks. Järgnevalt esitame nimekirja funktsionaalsustest, mis on kasutajale võimaldatud pärast graafide sisse laadimist:

- eri failides olevate graafide vahel liikumine, klahvi- või nupuvajutusega;
- graafi tippude liigutamine;
- graafi servade kaarjaks muutmine;
- graafi kaarjate servade sirgeks taastamine;
- graafide salvestamine rakenduse failiformaadis;
- graafide väljastamine PDF-vormingus.

Vastavalt peatükis 1 kirjeldatud eesmärkidele, lubab valminud rakendus sisse laadida mitu graafi mille vahel saab rakenduse tööajal liikuda kasutades selleks kas nuppe "<–" ja "–>", mis asuvad akna paremal all nurgas, või kasutades klaviatuuri paremale ja vasakule klahve. Kui laadida sisse graafe DIMACS-vormingus, on kasutajal võimalus valida mitu faili, kuna DIMACS-vorming toetab ainult ühe graafi kirjeldust faili kohta. Kui aga laadida graafe sisse rakenduse enda failiformaadis salvestatud failist, saab valida ainult ühe faili, kuna rakenduse enda failiformaat toetab mitme graafi kirjeldust ühes failis.

3.2 Sisendandmed ja nende töötlus

Graafide visualiseerimise rakendusel on kahte erinevat formaati sisendandmeid. Esimeseks sisendandmete tüübiks on DIMACS-vormingus salvestatud graafi kirjeldused. DIMACS-vorming graafi probleemide kirjeldamiseks on standardiseeritud viis, kuidas märkida üles graafe andmestruktuurina. See formaat sisaldab kolme eri tüüpi ridasid. Järgnevalt esitatakse kõik selle formaadi rea tüübid: [Ped01]

- c <sõne>
- p edge <n> <m>
- e <p1> <p2>

Esimene reatüüp on kommentaar, mis lubab faili sisestada teksti, mida arvutid peaksid ignoreerima. Selle rea tunnuseks on üksik "c" täht millele järgneb tühik, pärast mida tuleb kommentaari sõne. Teiseks reatüübiks on probleemi kirjeldus. Probleemi kirjeldus annab probleemist lühiülevaate. Antud rakenduse puhul annab see rida ülevaate graafist. Probleemi kirjelduse reatunnuseks on täht "p" millele järgneb tühik, siis sõna "edge", pärast mida tuleb tühikutega eraldatult kaks arvulist väärtust, kus esimene väärtus näitab graafi tippude arvu ja teine graafi servade arvu. Kolmas reatüüp selles formaadis on külje kirjeldus. Külje kirjelduse tunnuseks on täht "e" millele järgneb kaks arvulist väärtust, mis on tühikutega eraldatud ja mis tähistavad, millised tipud graafis on omavahel ühendatud.
Kuna kõik DIMACS-vormingus graafi kirjelduse rea tüübid sisaldavad endas kindlatel kohtadel tühikuid ning märksõnu, on võimalik töödelda neid faile efektiivselt, kasutades selleks põhimõtet jaotada sisseloetud sõned tühikute koha pealt massiiviks. See lubab kiiresti töödelda faili ridu, saada kätte vajaminevaid väärtusi ning leida võimalikke vigu faili sisu vastavusest formaadile. Rakenduse teiseks sisendandmete liigiks on graafide visualiseerimise rakenduse enda failiformaadis graafide kirjeldused. Nii nagu selgub peatükis 3.3 on ka rakenduse enda failiformaat range kindlate märksõnade ja väärtuste, aga ka faili struktuuri suhtes. Tänu sellele saab kasutada ka rakenduse enda salvestuste formaadi sisselugemisel samu põhimõtteid andmete töötlemisel nagu kasutatakse DIMACS-vormingus graafi kirjelduste puhul.

3.3 Failiformaat graafide salvestamiseks

Graafide visualiseerimise rakendus omab enda failiformaati andmete salvestamiseks. See failiformaat on range süntaksiga ning on disainitud selleks, et saaks korraga ühes failis hoida mitut graafi koos nende tippude, servade ja nimedega. Järgnevalt esitame failistruktuuri näidise:

<sõne>
vertices
<x> <y> <sõne>
edges
<p1> <p2> <x> <y>
---=---

Iga graafi kirjeldus algab sõnega, mis määrab graafi nime. Pärast graafi nime tuleb uuel real märksõna "vertices", mis tähistab tippude loetelu algust. Tipu kirjeldus sisaldab kolme elementi. Esimesed kaks väärtust on x- ja y-koordinaadid ning kolmas element on sõne, mis on tipu nimeks. Pärast tippude loetelu tuleb märksõna "edges", mis tähistab servade loetelu algust. Serva kirjeldus sisaldab valikuliselt kas kaks või neli elementi sõltuvalt sellest, kas serv on kaarjal kujul või mitte. Esimesed kaks elementi on alati külje alg- ja lõpptipu nimed. Valikuliselt võib lisada juurde kaks elementi, milleks on x- ja y-koordinaadid, läbi mille joonistatakse külg, kasutades selleks Bézier'i kurvi. [GEF02] Kui fail peab sisaldama rohkem kui ühte graafi, saab graafe omavahel eraldada kasutades selleks märgendite jada "---=---". See märgendite jada on range ning ei ole muudetav. Kui fail sisaldab ainult ühte graafi, võib eraldamise märgise ära jätta.

3.4 Graafide esitamine ekraanil

Graafide visualiseerimise rakendus kasutab graafide esitamiseks ekraanil kooridnaatteljestikku, kus x-telg suureneb vasakult paremale ja y-telg suureneb ülevalt alla. Rakendus kasutab tippude ja servade meelespidamiseks objektorienteeritud programmeerimise põhimõtteid. Rakendus hoiab massiivi graafi objektidest, mis omakorda sisaldavad massiive tippude ja servade objektide jaoks. Tipuobjekt omab x- ja y-koordinaati, tänu millele oskab rakendus antud punkti joonistada ekraanile. Servaobjektid ei sisalda koordinaate algus- ja lõpppunkti jaoks, vaid ainult tippude nimesid mida serv ühendab. See muudab tippude asukoha muutmise mugavaks, sest serv on seotud tipuga, mitte

positsiooniga ekraanil. Rakendus kuvab ekraanile korraga ainult ühe graafi. See tähendab, et korraga on aktiivne ainult üks graaf graafiobjektide massiivist ning ainult seda graafi tuleb joonistada. Kui aga kasutaja peaks mõne tipu asukohta muutma, muutub selle tipu asukoht kõikidel graafidel. Sama kehtib ka külgede kaarjaks muutmise kohta.

3.5 Graafide salvestamine PDF-vormingusse

Graafide visualiseerimise rakendus kasutab graafide PDF-vormingusse muundamiseks dokumentide ettevalmistussüsteemi LaTeX. LaTeX-süsteem toetab väga suurt kogust teeke, mis lihtsustavad erinevaid tegevusi dokumentide loomisel. Antud rakenduse puhul graafi joonistamist dokumendi sisse. Ka meenutab LaTeX-süntaks tugevalt programmeerimise keelt, mis lubab kirjutada funktsioone selleks, et genereerida kindlaid lõike dokumendist automaatselt. Lisaks lubab LaTeX-süsteem kirjutada faili, mis kirjeldab dokumenti ja tema sisu enne, kui see dokument reaalselt luuakse. See muudab LaTeX-süsteemi ideaalseks programmisiseseks kasutamiseks, et genereerida automaatselt PDF-vormingus dokumente. [Lat]

3.5.1 LaTeX-dokumentide ettevalmistussüsteem

LaTeX on dokumentide ettevalmistussüsteem, mis sisaldab endas märgenduskeelt ja TeX-programmi. LaTeX-süsteemi sügavamaks mõtteks, on eraldada dokumendi kujundus ja sisu. Selleks on LaTeX-süsteemi kasutajale antud suur kogus erinevaid teeke, mis sisaldavad käske, et muuta dokumendi välimuse kirjeldamine võimalikult lihtsaks. Kõik LaTeX-süsteemi käsud algavad langeva kaldkriipsuga ja sisaldavad endas mingit sõna, mis on tavaliselt loogiline dokumendi struktuuri suhtes nagu näiteks "\title{Minu teema}" või "\documentclass[12pt]{article}". [Lat] Suuremate seletusteta on võimalik aru saada, mida need käsud dokumendi loomise kontekstis tähendavad. Lisaks on võimalik LaTeX-süsteemi abil kujutada erinevaid keerulisi valemeid. Näiteks

```
X = \frac{m_{0}} {\sqrt{m + 4 * y ^ 2}}
```

on võrdväärne dokumendisisese valemiga

$$X = \frac{m_0}{\sqrt{m+4*y^2}}.$$

Sellepärast on LaTeX-süsteem populaarne ka akadeemilistes ringkondades, eriti informaatikute ja matemaatikute seas, kus erinevad dokumendid võivad sisal-

dada keerulisi jooniseid, diagramme ja valemeid.

3.5.2 Graafi paigutamine dokumenti

Graafi paigutamine ekraanilt, mis on resolutsioonist ja pikslite tihedusest sõltuv pind, PDF-dokumenti, mis eelnevalt nimetatud parameetritest ei sõltu, vajab natukene eeltööd. Selleks tuleb esialgu leida mis süsteemi põhimõttel joonistab LaTeX-süsteem mitme geomeetrilise kujundiga jooniseid dokumenti. Graafide visualiseerimise rakenduse tegemisel otsustas autor LaTeX-süsteemi teegi TikZ kasuks, kuna see sisaldab otseseid käske graafi tippude ja servade joonistamiseks. [Tik] Lisaks sellele, lubab TikZ joonistada graafe väga lihtsale tippude kaugusest sõltuvas koordinaatteljestikus ning toetab suunatud graafe ja graafi elementide disaini muutmist. Tippude kaugusest sõltuv koordinaatteljestik tähendab, et kui panna dokumendis paika üks graafi tipp, siis kõik ülejäänud tipud ja nende kaugused joonistatakse suhteliselt sellest kõige esimesest tipust. Pärast katse-eksituse meetodi kasutamist leidis autor, et maksimaalne kaugus, mis tohib olla kahe tipu vahel ilma, et graafi tipud ületaksid A4 ($210mm \times 297mm$) paberile määratud ääriste piire, oli 19 cm x-teljel ja 27.5 cm y-teljel.
Sellega on eeltööd tehtud. Edasi kirjeldame lähemalt LaTeX-süsteemi käske mille abil on võimalik luua graafe. Tänu TikZ teegile on võimalik lihtsasti luua dokumente, mis sisaldavad graafi kujutisi. Tippude joonistamiseks annab TikZ kasutajale käsu: [Tik]

\node (<p1>) at (<x>,<y>) {<tipu nimi>};

kus <p1> on tipu identifikaator, mis peab olema unikaalne. Märgised <x> ja <y> on punkti koordinaadid koordinaatteljestikus ja <tipu nimi> on sõne mis ütleb mida peaks näitama dokumendis tipu sees. Servade joonistamiseks on aga käsk: [Tik]

\draw (<p1>) to (<p2>);

kus <p1> ja <p2> on indentifikaatorite väärtused tippudel, mis tuleb omavahel joonega ühedada.
Järgmisena kirjeldame põhimõtet mille alusel on võimalik luua funktsioon, mis automaatselt genereerib dokumendi, mille sisse on joonistatud graaf. Selle jaoks, et oleks võimalik graafi joonistada dokumenti, mis kasutab mõõtühikuna meetermõõdustikku, ekraanilt, mis kasutab mõõtühikuna piksleid, mis ei ole kunagi standardiseeritud suurusega, tuleb kasutada väärtuste muundamisel

suhtelisi kaugusi. Selle asemel, et kasutada graafi joonistamisel punktide asukohti ekraanil, kasutame rakenduses hoopis tippudevahelisi kaugusi. Esimese asjana võetakse ekraanil olevast graafist täiesti suvaline punkt, millest saab "nullpunkt". Nullpunkti nimi tuleneb faktist, et tema koordinaadid loodavas dokumendis saavad olema $x = 0$ ja $y = 0$. Järgmisena käiakse läbi kõik tippude paarid antud graafis ja leitakse maksimaalne vahe kahe punkti vahel nii x-teljel kui ka y-teljel. Nimetame neid väärtusi x_{max} ja y_{max}. Samal ajal leitakse ka kõige väiksem kaugus kahe punkti vahel l_{min}, mida arvutatakse kasutades selleks Pythagorase teoreemi, antud näite puhul $l = \sqrt{x^2 + y^2}$. Kui need kolm väärtust on leitud, tuleb hinnata, kas graafi on võimalik joonistada dokumenti, ilma graafi mõõtmeid vähendamata. Selleks, et hinnata kas graafi mõõtmeid tuleb vähendada, tuleb anda tulevasele joonisele, mis on meetermõõdustikus, algsuurus. Nimetame *algsuuruseks* kõige väiksemat kahe punkti vahelist kaugust, mis joonisel on olemas. Graafide visualiseerimise rakenduses valiti selle vahe suuruseks 1 cm. Kuna graafi tipud ei ole punktid, vaid ringid, mis omavad raadiust, tuleb sellesse algsuurusesse sisse arvestada ka kahe punkti raadius, selleks et joone pikkus dokumendis oleks tõesti 1 cm. Antud rakenduses valiti graafi tipu raadiuseks 0.3 cm, mis tähendab, et algsuuruseks l_{alg} on uuel dokumendil 1.6 cm (1cm + 2 * 0.3cm).
Kasutades lihtsat matemaatikast tuntud ristkorrutist, on võimalik leida punktide asukohad loodavas dokumendis, kasutades selleks ainult ekraanil olevate tippude asukohti. Ristkorrutis kasutab ära murru omadust väljendada kahe arvu vahelist suhet. Luues kahest murrust võrrandi, on võimalik arvutada sama kauguste suhe erineva mõõdusüsteemi jaoks. Valemina väljendatult on see

$$\frac{a}{b} = \frac{c}{d} \rightarrow d = \frac{b*c}{a} \ .$$

Kasutades seda põhimõtet, on võimalik arvutada, kas graaf ületab maksimaalse dokumendi pikkuse või laiuse, kui algsuuruseks on l_{alg}. Selle jaoks teeme kaks arvutust, nii x-telje kui y-telje jaoks ning hindame kas tulemus on suurem kui seatud piirmäärad.

$$\frac{x_{max}*l_{alg}}{l_{min}} > 19 \ ,$$

$$\frac{y_{max}*l_{alg}}{l_{min}} > 27.5 \ .$$

Kui mõlemad võrratused on väärad, siis on võimalik graaf paigutada dokumenti ilma algsuurust vähendamata. Selleks tuleb iga tipu x- ja y-koordinaat panna eelpool toodud funktsioonidesse, vastavalt x_{max} ja y_{max} asemele ning

vastuseid kasutada argumentidena ülalpool kirjeldatud LaTeX-süsteemi käsus. Kui vähemalt üks nendest võrratustest on tõene, tuleb vähendada algsuurust l_{alg} , et graaf mahuks ära kogu ulatuses ilma tippudevaheliste kauguste suhteid rikkumata. Selle jaoks saab taas kasutada ristkorrutist, kuid seekord eesmärgiga leida l_{alg} suurust. Taaskord tuleb arvutus läbi teha nii x- kui y-teljel ning valida, tulemustest väiksem väärtus.

$$l_{alg1} = \frac{l_{min}*19}{x_{max}} \ ,$$

$$l_{alg2} = \frac{l_{min}*27.5}{y_{max}} \ .$$

Kui l_{alg} väärtus on korrigeeritud, ei riku graafi kujutis enam dokumendile seatud piire, mis tähendab, et ülalpool esitatud võrratused ei ole enam tõesed. Järelikult saab jätkata graafi tippude koordinaatide teisendamist uuele mõõdusüsteemile.
Graafide visualiseerimise rakenduses rakendatakse sama põhimõtet igale graafile, mida tahetakse väljastada PDF-dokumenti. Kasutades koordinaatväärtusi uues mõõdusüsteemis luuakse TeX-fail, mis sisaldab kogu dokumendi kirjeldust sealhulgas graafide kirjeldusi. Viimase asjana, kasutatakse programmeerimiskeele võimalusi, et kutsuda välja käsureal LaTeX-süsteemi käsk (autori poolt kasutatava LaTeX-süsteemi realisatsiooni puhul oli see käsk *pdflatex*), mis koostab TeX-failis sisalduvatest käskudest ja tekstist PDF-faili.

3.6 Rakenduse muud kasutusvõimalused

Kuigi graafide visualiseerimise rakendus on ette nähtud täitma antud töö eesmärke, on seda võimalik rakendada muudel viisidel. Järgnevalt toome mõned näited, mida on võimalik selle rakendusega teha, mida otseselt eesmärkides ei nõutud.
Kuigi peatükis 1 on eesmärgiks seatud, et rakendus peab võimaldama mitme graafi korraga sisselaadimist, ei ole see range tingimus. Rakendus võimaldab sisse laadida ka ainult ühte graafi. Selline funktsionaalsus lubab kasutajal muuta ühte kindlat graafi, või kontrollida ühe graafi mõnda omadust.
Lisaks võib laiendada mitme graafi sisselaadimise tingimust ka teistpidi. Kuigi peatükis 1 on eesmärgiks seatud, et rakendus peab võimaldama sisse laadida mitu graafi milles on täpselt sama palju tippe, ei ole ka see tingimus range. Rakendus lubab sisse laadida mitu graafi suvaliste tippude arvuga. Konfliktid lahendatakse tipu nime põhiselt. See tähendab, et kui mitmel graafil on sama nimega tipp, siis on see tipp kõikidel graafidel samas kohas. Tippe mis esinevad

osades graafides aga mitte hetkel aktiivses graafis, ei näidata. Selline funktsionaalsus lubab kasutajal sisse laadida erinevate graafide kogumiku, mis ei pruugi olla otseselt seotud, kuid mida soovitakse ikkagi PDF-kujul väljastada.

4 Rakenduse edasiarendamise võimalused

Kuigi graafide visualiseerimise rakendus on tervik ning täidab kõiki nõudeid, mis püstitati peatükis 1 on ka sellel rakendusel palju edasiarendamise võimalusi. Järgnevalt toome välja näiteid sellest, kuidas autori arvates oleks võimalik rakendust paremaks teha.

Esimesena võiks käsitleda ebaefektiivsustest rakenduse lähtekoodis. Autori arvamuseks on, et rakenduse lähtekoodi oleks võimalik suuresti optimeerida ning refaktoreerida. Lugedes lähtekoodi on võimalik leida mitu kohta, kus sama koodilõiku on kasutatud korduvalt erinevates kohtades. See on koodi struktuuri vaatepunktist halb ning parem oleks nendest koodilõikudest teha eraldiseisvad funktsioonid, mis vähendaks koodi ridade arvu märgatavalt.

Koodi struktuuri uurides võib ka märgata, et kindlad osad rakendusest moodustavad nii-öelda omaette tervikuid ning nende tervikute vahele on loodud lisa kood spetsiaalselt selle jaoks, et edastada infot ühest kohast teise. Ka selline lähenemine ei ole lähtekoodi struktuuri mõttes efektiivne. Koodi osad võivad moodustada omaette tervikuid, kuid nad ei tohiks vajada veel lisatööd selle jaoks, et transportida informatsiooni ühest kohast teise. Niisiis, kui jätta programmi loogika samaks ning refaktoreerida lähtekoodi üldist struktuuri, saaks rakenduse jõudlust ja lähtekoodi loetavust veel omajagu parandada.

Järgmisena käsitleme rakenduse loogika parendamisest. Autori arvates on rakenduse loogikas edasiarendamise võimalusi sisseloetud andmete töötlemisel ja graafi tasandilisuse tõestamisel. Praeguse seisuga toimub sisseloetud andmete töötlemine väga naiivsel ning lihtsal tasemel. Rakendus suudab ära tunda väga palju erinevaid vigu, mis on seotud sisseloetud faili struktuuriga, ent tuleb nentida, et kindlasti leidub selliseid sisendeid, mille peale rakendus võib anda valenegatiivseid tulemusi, mis võivad kahjustada rakenduse korralikku töötamist. Niisiis võiks lahendusena välja pakkuda, et sisseloetud failide andmete töötlemise võiks muuta paremaks rakendades selleks mõnda automaatset olekumasinat, mis on seadistatud töötlema sõltuvalt sisseloetud faili tüübist kas DIMACS-vormingusse või rakenduse enda failiformaati. See muudaks rakenduse rohkem vastupidavamaks erinevate pahalaste suhtes, kes üritavad rakendusele ette anda faile, mis on disainitud selleks, et häirida rakenduse tööd.

Graafide visualiseerimise rakenduse graafide tasandilisuse tõestamise algoritm on ka üks kohtadest, kus oleks võimalik rakendust edasi arendada. Kuna graafide tasandilisuse tõestamise algoritm on väga keeruline, on see ka koht, kus realisatsioonis võib kõige rohkem vigu esineda. Nagu räägiti peatükis 1, ei ole rakenduse eesmärgiks igal korral tõestada graafi tasandilisust, vaid pigem

pakkuda kasutajale informatsiooni selle kohta, kas graaf võib olla tasandiline ja kui rakendus sellega hakkama saab, anda kasutajale graafi kujutis, mis võiks olla abistav tasandilise kujutise leidmisel. Antud teema puhul on autori soovituseks katse-eksituse meetod. Rakendusest tuleks läbi lasta suur kogus erinevaid graafe, mille tasandilisuse omaduse kehtivus on teada ning kontrollida, kas rakendus annab antud graafi kohta valeinformatsiooni. Kui rakendus peaks hindama graafi valesti, on see märk sellest, et kusagil lähtekoodis on võimalik teha parandusi selleks, et rakendus sellist viga enam ei teeks.

Viimasena käsitleme rakenduse edasiarendamisest funktsioonide lisamise teel. Autori arvamus on, et graafide visualiseerimise rakendust saaks paremaks teha lisades sellele juurde graafide loomise funktsionaalsus. Graafide loomine ilma graafi kirjeldavat faili sisse lugemata, oleks väga kasulik funktsionaalsus kasutajale. Selleks tuleks rakendusele lisada tööriba, mis sisaldab endas tippude ja servade loomise ning ka kustutamise tööriistu. Selline funktsionaalsus oleks kasulik, kui graafi jaoks, mida kasutaja tahab kuvada või välja trükkida, ei ole olemas vastavat faili. See hoiaks kokku palju aega sellega, et kasutaja ei peaks hakkama kirjutama sellist faili, ning sellega, et ta ei peaks muretsema sisseloetava faili struktuuri pärast. Alamhulgana sellest funktsionaalsusest võiks ka lisada juba sisseloetud graafidele tippude ja servade lisamise ning kustutamise. Selline funktsionaalsus annaks kasutajale võimaluse muuta juba sisseloetud graafi. See taaskord hoiaks kokku palju kasutaja aega, kes ei peaks siis enam avama graafi kirjelduse faili ning käsitsi muutma sealolevaid väärtusi selleks, et mainitud operatsioone sooritada.

Igale rakendusele on võimalik teha parandusi sõltuvalt programmeerija vaadetest ja ideedest. Need olid osad näited, mida autor pidas vajalikuks eraldi välja tuua kõigile, kes plaanivad võib-olla tulevikus teha tööd graafide visualiseerimise rakenduse kallal.

5 Hinnang graafide visualiseerimise rakendusele

Graafide visualiseerimise rakendusele antakse hinnang kahes osas. Esimeseks osaks on rakenduse vastavus töös esitatud eesmärkidele ning teiseks osaks on rakenduse jõudlus praktilises mõttes.

5.1 Rakenduse vastavus eesmärkidele

Peatükis 1.2 määratud eesmärkide järgi peab olema graafide visualiseerimise rakenduses võimalik graafi tippude liigutamine. See funktsionaalsus on realiseeritud ning töötab kasutades selleks arvuti hiirt. Sellest funktsionaalsusest on rohkem juttu peatükis 3.1. Järgmisena pidi valmis rakendus omama funktsionaalsust külgede kaarjaks muutmisest. Ka see funktsionaalsus on realiseeritud ning selle kohta on lisa informatsiooni peatükis 3.1. Viimasena oli rakendusele nõue, et kõik sisse laetud graafid peab olema võimalik väljastada PDF-vormingus faili. See funktsionaalsus on olemas ning selle kohta on pikemalt kirjutatud peatükis 3.5. Lisaks eelnevatele lisafunktsionaalsustele, pidi rakendus realiseerima põhifunktsionaalsusena graafide sisselaadimise ning tasandilisuse lähedase kujutise leidmise. Ka põhifunktsionaalsused on graafide visualiseerijas olemas ning töökorras. Põhifunktsionaalsuste kohta võib pikemalt lugeda peatükkidest 2 ja 3. Kuna graafide visualiseerimise rakendus täidab kõik peatükis 1.2 esitatud nõudmised võib öelda, et rakendus on õigesti realiseeritud.

5.2 Jõudlus

Kuna jõudluse osas ei tehtud rakendusele mingeid nõudmisi, on selle osa hinnang subjektiivne. Siiski arvab autor, et isegi kui võtta arvesse edasiarendamise võimalusi, mida on kirjeldatud peatükis 4, võib siiski väita, et rakenduse jõudlus on hea. Jõudluse hinnangu põhjenduseks võib öelda, et kuna arvuti ekraan on piiratud ala, siis ei ole võimalik luua nii suurt graafi, mis aeglustaks arvuti tööd piisavalt, et tekiksid ülemäära pikad ooteajad, ilma et see graaf oleks loetamatu arvuti ekraani pinnalt, isegi kui tasandiline kujutis leidub.

6 Kokkuvõte

Graafide visualiseerimine on ülesanne mida saab lahendada kasutades selleks erineva raskusastmega ideid ja algoritme. On võimalik luua lihtsaid rakendusi, mis visualiseerivad graafe ning on võimalik luua äärmiselt keerukaid rakendusi, mis suudavad lahendada tasandilisuse tõestusi ning graafi kujutiste loomisi.
Antud töö eesmärgiks oli luua rakendus, mis suudaks kujutada graafi ekraanil, kuid aidata ka kasutajat võimalikult palju tasandilisuse probleemi lahendamisel ning tasandilise kujutise leidmisel.
Töös on kirjeldatud radade lisamise algoritmi graafi tasandilisuse tõestamiseks ning graafide esitamise põhimõtet kasutades selleks tasandilisuse tõestuse andmestruktuure ja andmeid. Lisaks on kirjeldatud põhimõtted, mille alusel on loodud rakendus graafide kuvamiseks.
Lõputöö tulemusena valmis rakendus, mis on võimeline kuvama suvalist graafi. Lisaks sellele oskab rakendus kasutajat aidata ülesandes leida graafi tasandiline kujutis ning annab kasutajale väikese hulga erinevaid tööriistu graafi muutmiseks. Ka lubab rakendus graafe väljastada trükkimiseks sobivas PDF-vormingus.
Töö analüüsivas osas hinnatakse graafide visualiseerimise rakenduse jõudlust ning efektiivsust ja antakse soovitusi, kuidas oleks võimalik seda rakendust edasi arendada.

Kasutatud kirjandus ja allikad

[AB03] P. Laud J. Villemson A. Buldas. *Graafid*. Tartu Ülikooli Kirjastus, 2003. http://research.cyber.ee/~peeter/teaching/graafid03s/graafid.pdf Viimati vaadatud 2015.04.22.

[AP11] Mati Tombak Ahti Peder. Finding the description of structure by counting method: a case study. *SOFSEM 2011: Theory and Practice of Computer Science*, 2011.

[BiCa] biconnected graph. http://xlinux.nist.gov/dads//HTML/biconnectedGraph.html Viimati vaadatud 2015.04.22.

[BiCb] Kaksik-sidus graaf. http://www.geeksforgeeks.org/wp-content/uploads/Biconnected1.png Viimati vaadatud 2015.04.22.

[fDMD93] Center for Discrete Mathematics and Theoretical Computer Science (DIMACS). Satisfiability suggested format, 1993. 8 pp.

[GEF02] Myung-Soo Kim Gerald E. Farin, Josef Hoschek. *Handbook of Computer Aided Geometric Design*. Elsevier Science B.V., 2002. http://books.google.ee/books?id=0SV5G8fgxLoC&pg=PA4&redir_esc=y#v=onepage&q&f=false Viimati vaadatud 2015.04.22.

[JH73] R. Tarjan J. Hopcroft. Efficient algorithms for graph manipulation. *Communications of the ACM*, 1973.

[JH74] R. Tarjan J. Hopcroft. Efficient planarity testing. *Journal of the Association for Computing Machinery*, 1974.

[K33] K3,3 graaf. Kirjeldus pärineb raamatust "Graafid" [AB03], illustratsioon veebilehelt: http://upload.wikimedia.org/wikipedia/commons/thumb/f/f3/Biclique_K_3_3.svg/200px-Biclique_K_3_3.svg.png Viimati vaadatud 2015.04.22.

[K5] K5 graaf. Kirjeldus pärineb raamatust "Graafid" [AB03], illustratsioon veebilehelt: http://upload.wikimedia.org/wikipedia/commons/thumb/c/cf/Complete_graph_K5.svg/612px-Complete_graph_K5.svg.png Viimati vaadatud 2015.04.22.

[Kug] Mirko Kugelmeier. Efficient planarity testing. http://tcs.rwth-aachen.de/lehre/Graphentheorie/WS2013/Mirko_Kugelmeier.pdf. Slaidid. Viimati vaadatud 2015.04.22.

[Lat] Latex koduleheküIg. http://latex-project.org/intro.html Viimati vaadatud 2015.04.22.

[Mit] Mitte kaksik-sidus graaf. http://www.geeksforgeeks.org/wp-content/uploads/Biconnected4.png Viimati vaadatud 2015.04.22.

[Ped01] Ahti Peder. Metamuutujatega loogikavalemite translaator. Master's thesis, Tartu Ülikool, 2001.

[Pla] Lecture notes on planarity testing and construction of planar embedding. http://www.csd.uoc.gr/~hy583/papers/ch15.pdf Viimati vaadatud 2015.04.22.

[Tay08] Martyn G. Taylor. *Planarity testing by path addition.* PhD thesis, The University of Kent at Canterbury, 2008. https://archive.org/stream/PlanarityTestingByPathAddition/PlanarityTestingByPathAddition-MartynGTaylor-2012-05-30#page/n0/mode/1up Viimati vaadatud 2015.04.22.

[Tik] *The TikZ and PGF Packages Manual for version 2.10-cvs.* http://www.texample.net/media/pgf/builds/pgfmanualCVS2012-11-04.pdf Viimati vaadatud 2015.04.22.

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Sander Tiganik (sünnikuupäev: 08.05.1993),

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose:
   Kaksiksidusate graafide visualiseerija
   mille juhendaja on Ahti Peder,

1.1 reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace'i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartu, **14.05.2015**